Global Information Assurance Certification Paper

GIAC Intrusion Detection Analysis Practical Exam

J.D. Baldwin

San José session, May 2000

The following are analyses of ten network intrusion detections, completed in fulfillment of the GIAC CIA certification program.

Detect 1

```
12:56:34.729690 cc105035-a.hwrd1.md.home.com.3211 > xxx.yyy.240.4.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359532[|tcp]> (DF)
12:58:33.340938 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.5.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359337[|tcp]> (DF)
12:58:33.341215 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.6.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359210[|tcp]> (DF)
12:58:33.341946 cc105035-a.hwrd1.md.home.com.3412 > xxx.yyy.240.7.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359449[|tcp]> (DF)
12:58:33.345393 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.9.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359672[|tcp]> (DF)
```

[more sequential probes deleted]

```
12:58:35.381880 cc105035-a.hwrd1.md.home.com.3313 > xxx.yyy.240.22.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359336[|tcp]> (DF)
12:58:35.383354 cc105035-a.hwrd1.md.home.com.3412 > xxx.yyy.240.23.domain: S 1901647080:1901647080(0)
        win 32120 <mss 1460,sackOK,timestamp 142359970[|tcp]> (DF)
12:58:35.383402 xxx.yyy.240.23.domain > cc105035-a.hwrd1.md.home.com.3412: R 0:0(0)
        ack 1901647081 win 0 (DF)
12:58:35.385689 cc105035-a.hwrd1.md.home.com.3211 > xxx.yyy.240.24.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359228[|tcp]> (DF)
12:58:35.387273 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.25.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359117[|tcp]> (DF)
```

[more sequential probes deleted]

```
12:58:42.217282 cc105035-a.hwrd1.md.home.com.3331 > xxx.yyy.240.5.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359872[|tcp]> (DF)
12:58:42.217818 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.6.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142360647[|tcp]> (DF)
12:58:42.217975 cc105035-a.hwrd1.md.home.com.3211 > xxx.yyy.240.7.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359044[|tcp]> (DF)
12:58:42.220358 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.9.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359633[|tcp]> (DF)
```

[more sequential probes deleted]

```
12:58:49.273290 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.119.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359247[|tcp]> (DF)
12:58:49.274882 cc105035-a.hwrd1.md.home.com.3313 > xxx.yyy.240.120.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359124[|tcp]> (DF)
```

1. Source of trace:

Unused (as yet) Internet connection for new data center of a large midwestern company. xxx.yyy.240 is the network of the outside-the-firewall DMZ.

2. Detect was generated by:

tcpdump (no filtering) running on a UNIX machine attached to a switch just inside the Internet router (but outside of the firewall). Traffic on this net is low enough that tcpdump files may be examined by hand once a few authorized hosts are filtered out.

3. Probability that the source address was spoofed:

Low. This attack probably wouldn't be much use in its observed form if the source address were spoofed.

However, the machine identified (cc105035-a.hwrd1.md.home.com) did not respond on any port to an nmap scan. The trace was caught almost immediately, and attempts to contact the machine were tried right away. There are no other indications that this might have been a spoofed address.

4. Description of attack:

The host cc105035-a.hwrd1.md.home.com is attempting to connect ("S" indicates the SYN flag) to TCP port 53 of IP addresses on this subnet, sequentially. The packets themselves contain no data, as indicated by the (0) after the sequence numbers. The narrow and repeating set of source ports used (3095, 3211, 3412, 3331) is an indication that these packets are crafted.

This is basically a host sweep / mapping attempt, looking for hosts running named that will respond to TCP connection attempts on the DNS port, 53.

Note that .23 responded, at 12:58:35, with an immediate reset, since it is live and does not run named. (Other live hosts outside the firewall did the same.) Almost none of the other addresses probed map to live hosts. Interestingly, xxx.yyy.240.8 is a live host, and a very important one; it is the firewall protecting this gateway, yet it was skipped in both scans. The host at .23 is of course live as well, but was very new on this subnet. This is an indication that the attacker may already have some information about the machines on this net. Perhaps he has been here before and chose to examine only ports that didn't produce hits in an earlier scan.

5. Attack mechanism:

This is a host sweep for DNS servers, but (unlike Detect 2 below), no attempt has been made to conceal or obfuscate the requests.

Once found, machines responding to the TCP connection attempt on port 53 will either receive a DNS zone transfer request, or be attacked with buffer overflow or other malicious attacks directed against various versions of BIND or Windows DNS server (for DOS or system compromise). Or the responding hosts may simply be logged in a database so that they may be subjected to these sorts of attacks more efficiently later.

6. Correlations:

Scans like this are seen commonly in mapping attempts. This sort of thing was covered extensively during the May, 2000 SANS course in San José and is addressed in SANS course guide 2.2, *Intrusion Detection and Packet Filtering: How It Really Works*, by Vicki Irwin and Hal Pomeranz.

7. Evidence of active targeting:

The crafted packets (discussed above) and the sequential nature of the sweep are clear indications of active targeting.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	5	A firewall exists on this subnet, although other machines are much less critical.
Lethality	1	Probably just a mapping attempt; no real information is available about the network even if the attacker should happen across a DNS server. (See "Defensive recommendation," below.)
Countermeasures	4	The hosts on this net are well maintained and recently patched. None of them provide DNS services.
Network CM	1	While a strong firewall protects the internal network, this subnet is exposed directly to the Internet.
TOTAL SEVERITY	1	(5 + 1) - (4 + 1)

9. Defensive recommendation:

There is nothing really to be done here. This attack was directed to a net outside the firewall. A review of firewall rules showed that all external DNS queries are discarded (as the company hosts a DNS server specifically for outside queries only). The machines outside this firewall are non mission-critical and have been locked down pretty well (unnecessary services turned off, etc.).

10. Multiple choice test question:

Given the following trace:

```
12:56:34.729690 cc105035-a.hwrd1.md.home.com.3211 > xxx.yyy.240.4.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359532[|tcp]> (DF)
```

```
12:58:33.340938 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.5.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359337[|tcp]> (DF)
12:58:33.341215 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.6.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359210[|tcp]> (DF)
12:58:33.341946 cc105035-a.hwrd1.md.home.com.3412 > xxx.yyy.240.7.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359449[|tcp]> (DF)
12:58:33.345393 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.9.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359672[|tcp]> (DF)
12:58:35.381880 cc105035-a.hwrd1.md.home.com.3313 > xxx.yyy.240.22.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359336[|tcp]> (DF)
12:58:35.383354 cc105035-a.hwrd1.md.home.com.3412 > xxx.yyy.240.23.domain: S 1901647080:1901647080(0)
        win 32120 <mss 1460,sackOK,timestamp 142359970[|tcp]> (DF)
12:58:35.383402 xxx.yyy.240.23.domain > cc105035-a.hwrd1.md.home.com.3412: R 0:0(0)
        ack 1901647081 win 0 (DF)
12:58:35.385689 cc105035-a.hwrd1.md.home.com.3211 > xxx.yyy.240.24.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359228[|tcp]> (DF)
12:58:35.387273 cc105035-a.hwrd1.md.home.com.3095 > xxx.yyy.240.25.domain: S 1898969486:1898969486(0)
        win 32120 <mss 1460,sackOK,timestamp 142359117[|tcp]> (DF)
```

What is the indication that this is a directed attack?

a) Trying hosts sequentially for a response at port 53.

b) The same TCP sequence number for every originated packet.

c) Narrow, repeating range of source ports.

d) All of the above are indications of a directed attack.

Answer: d)

Detect 2

```
04:08:25.206247 63.202.5.68.53 > xxx.yyy.240.1.53: SF 1741485298:1741485298(0) win 1028 (ttl 30, id 39426)
04:08:25.225355 63.202.5.68.53 > xxx.yyy.240.2.53: SF 1741485298:1741485298(0) win 1028 (ttl 30, id 39426)
04:08:25.285875 arp who-has xxx.yyy.240.5 tell xxx.yyy.240.4
04:08:25.305341 63.202.5.68.53 > xxx.yyy.240.6.53: SF 1741485298:1741485298(0) win 1028 (ttl 30, id 39426)
04:08:25.324878 arp who-has xxx.yyy.240.7 tell xxx.yyy.240.4
04:08:25.344452 63.202.5.68.53 > xxx.yyy.240.8.53: SF 1741485298:1741485298(0) win 1028 (ttl 30, id 39426)
04:08:25.364590 arp who-has xxx.yyy.240.9 tell xxx.yyy.240.4
04:08:25.384515 arp who-has xxx.yyy.240.10 tell xxx.yyy.240.4
04:08:25.404459 arp who-has xxx.yyy.240.11 tell xxx.yyy.240.4
04:08:25.423801 arp who-has xxx.yyy.240.12 tell xxx.yyy.240.4
04:08:25.444947 arp who-has xxx.yyy.240.13 tell xxx.yyy.240.4
04:08:25.464094 arp who-has xxx.yyy.240.14 tell xxx.yyy.240.4
04:08:25.483241 arp who-has xxx.yyy.240.15 tell xxx.yyy.240.4
04:08:25.503390 arp who-has xxx.yyy.240.16 tell xxx.yyy.240.4
04:08:25.533906 arp who-has xxx.yyy.240.17 tell xxx.yyy.240.4
04:08:25.546675 arp who-has xxx.yyy.240.18 tell xxx.yyy.240.4
04:08:25.566439 arp who-has xxx.yyy.240.19 tell xxx.yyy.240.4
04:08:25.585970 arp who-has xxx.yyy.240.20 tell xxx.yyy.240.4
04:08:25.606712 arp who-has xxx.yyy.240.21 tell xxx.yyy.240.4
04:08:25.626261 arp who-has xxx.yyy.240.22 tell xxx.yyy.240.4
04:08:25.646510 63.202.5.68.53 > xxx.yyy.240.23.53: SF 1741485298:1741485298(0) win 1028 (ttl 30, id 39426)
04:08:25.646588 xxx.yyy.240.23.53 > 63.202.5.68.53: R 0:0(0) ack 1741485299 win 0 (DF) (ttl 64, id 22793)
04:08:25.646757 xxx.yyy.240.23.53 > 63.202.5.68.53: R 0:0(0) ack 1 win 0 (DF) (ttl 64, id 22793)
04:08:25.665574 arp who-has xxx.yyy.240.24 tell xxx.yyy.240.4
04:08:25.685900 arp who-has xxx.yyy.240.25 tell xxx.yyy.240.4
```

[... and further such arp who-has broadcasts up through the entire subnet.]

1. Source of trace:

Unused (as yet) Internet connection for new data center of a large midwestern company. xxx.yyy.240 is the network of the outside-the-firewall DMZ.

2. Detect was generated by:

tcpdump (no filtering) running on a UNIX machine attached to a switch just inside the Internet router (but outside of the firewall).

Traffic on this net is low enough that tcpdump files may be examined by hand once a few authorized hosts are filtered out.

3. Probability that the source address was spoofed:

There is no reason to believe that this address was spoofed. The machine identified (adsl-63-202-5-68.dsl.snfc21.pacbell.net) was active and responded to pings, though not to telnet attempts to ports 13, 23, 25 and 80. At the time this attack occurred, nmap was not readily available to me so I couldn't attempt to fingerprint the attacking host.

As the reverse lookup indicated, this address is from a netblock assigned to Pacific Bell, and further from a sub-netblock assigned to "Jack Hong" in San Francisco, CA. Judging from the hostname, it is a machine attached to a DSL line. No attempt was made to contact Mr. Hong in this case.

4. Description of attack:

This is a TCP SYN-FIN host sweep directed against port 53. The arp who-has packets are local traffic indicating that something (that the sniffing host can't see directly) is happening on the hosts for which arp information is requested.

This machine was on a switch that could not, unfortunately, be configured to repeat all traffic to its port. Therefore, the attacks against port 53 on all machines other than the ones where it was seen (.1, .6, .8, .23) are inferred from the fact that arp who-has requests were sent for these (non-existent) machines right in sequence and at the same time as the others. Because of the switch configuration, the actual attack packets were not repeated to this interface and were not captured.

5. Attack mechanism:

The host sweep is looking for DNS servers with the same basic intent as that described in Detect 1. The SYN FIN could be intended to conceal the sweep from some intrusion detection systems, or to slip past the rulebase of some firewall products. Also, the particular response a machine makes to an "impossible" packet like this can aid in fingerprinting: e.g., the fact that host .23 sent back a RESET-ACK might be useful information to the attacker in a later attack against this net.

6. Correlations:

Scans like this are seen often in mapping attempts. This sort of thing was covered extensively during the May, 2000 SANS course in San José and is addressed in SANS course guide 2.2, *Intrusion Detection and Packet Filtering: How It Really Works*, by Vicki Irwin and Hal Pomeranz. That sourcedescribes a SYN-FIN attack against port 53

It falls under the general heading of Cisco NDSB #3030. It is commonly seen on numerous ports; this particular one simply happens to be directed against port 53.

In the IDS database, it is called "SYN FIN Scan" or IDS198. It was contributed to that database by Marty Roesch. It is also listed as being directed against any port.

7. Evidence of active targeting:

Note, first, that the arp who-has requests indicate this sweep was tried on all empty IP addresses on the subnet, even though tcpdump didn't pick up the attack packets due to the configuration of the switch to which it is connected.

That aside, the sequential scanning of hosts in a short time and the SYN-FIN packets are clear indications of an active attack.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	5	A firewall exists on this subnet, although other machines are much less critical.
Lethality	1	Probably just a mapping attempt; no real information is available about the network even if the attacker should happen across a DNS server. (See the comments under "Defensive recommendation," below.)
Countermeasures	4	The hosts on this net are well maintained and recently patched. None of them provide DNS services.
Network CM	1	While a strong firewall protects the internal network, this subnet is exposed directly to the Internet.
TOTAL SEVERITY	1	(5 + 1) - (4 + 1)

9. Defensive recommendation:

This attack was directed to a net outside the firewall. A review of firewall rules showed that all SYN FIN packets are discarded, along with external DNS queries originating from outside the net, so no new defenses are needed there. Furthermore, no DNS servers exist on this net, and no DNS server external to the firewall contains information about the network inside the firewall. The machines outside this firewall are non mission-critical and have been locked down pretty well (unnecessary services turned off, patches up to date, etc.).

10. Multiple choice test question:

Given the following trace:

```
04:08:25.305341 63.202.5.68.53 > xxx.yyy.240.6.53: SF 1741485298:1741485298(0) win 1028 (ttl 30, id 39426)
```

Indications that this is a directed attack include:

a) DNS Zone Transfer attempt from unauthorized host.

b) Source port 53

c) Crafted packet, not found in "nature."

d) Bogus TTL value.

Answer: c)

Detect 3

```
14:23:55.188920 10.121.201.15.138 > xxx.yyy.197.148.138: udp 225 (ttl 127, id 22529)
14:23:55.458853 10.121.201.15.138 > xxx.yyy.198.3.138: udp 225 (ttl 127, id 23041)
14:23:56.628800 10.121.201.15.138 > xxx.yyy.175.249.138: udp 225 (ttl 127, id 25601)
14:23:56.848799 10.121.201.15.138 > xxx.yyy.147.7.138: udp 225 (ttl 127, id 26113)
14:23:57.313537 10.121.201.15.138 > xxx.yyy.171.248.138: udp 225 (ttl 127, id 27137)
14:23:57.858722 10.121.201.15.138 > xxx.yyy.197.15.138: udp 225 (ttl 127, id 28417)
14:23:58.220647 10.121.201.15.138 > xxx.yyy.197.148.138: udp 225 (ttl 127, id 29441)
14:23:58.446209 10.121.201.15.138 > xxx.yyy.198.3.138: udp 225 (ttl 127, id 29953)
14:23:59.677518 10.121.201.15.138 > xxx.yyy.175.249.138: udp 225 (ttl 127, id 32769)
14:23:59.809356 10.121.201.15.138 > xxx.yyy.147.7.138: udp 225 (ttl 127, id 33025)
14:24:00.492138 10.121.201.15.138 > xxx.yyy.171.248.138: udp 225 (ttl 127, id 34561)
14:24:00.716133 10.121.201.15.138 > xxx.yyy.197.15.138: udp 225 (ttl 127, id 35073)
14:24:01.219660 10.121.201.15.138 > xxx.yyy.197.148.138: udp 225 (ttl 127, id 36353)
14:24:01.442583 10.121.201.15.138 > xxx.yyy.198.3.138: udp 225 (ttl 127, id 36865)
14:24:02.672306 10.121.201.15.138 > xxx.yyy.175.249.138: udp 225 (ttl 127, id 39681)
14:24:02.810539 10.121.201.15.138 > xxx.yyy.147.7.138: udp 225 (ttl 127, id 39937)
14:24:03.439040 10.121.201.15.138 > xxx.yyy.171.248.138: udp 225 (ttl 127, id 41473)
14:24:03.673834 10.121.201.15.138 > xxx.yyy.197.15.138: udp 225 (ttl 127, id 41985)
```

[Note the break in time at this point.]

```
14:24:12.675320 10.121.201.15.138 > xxx.yyy.197.148.138: udp 212 (ttl 127, id 44289)
14:24:12.865980 10.121.201.15.138 > xxx.yyy.198.3.138: udp 212 (ttl 127, id 44801)
14:24:14.088906 10.121.201.15.138 > xxx.yyy.175.249.138: udp 212 (ttl 127, id 47617)
14:24:14.179716 10.121.201.15.138 > xxx.yyy.147.7.138: udp 212 (ttl 127, id 47873)
14:24:14.869453 10.121.201.15.138 > xxx.yyy.171.248.138: udp 212 (ttl 127, id 49409)
14:24:15.084072 10.121.201.15.138 > xxx.yyy.197.15.138: udp 212 (ttl 127, id 49921)
14:24:17.698649 10.121.201.15.138 > xxx.yyy.197.148.138: udp 212 (ttl 127, id 51457)
14:24:17.880341 10.121.201.15.138 > xxx.yyy.198.3.138: udp 212 (ttl 127, id 51969)
14:24:19.110143 10.121.201.15.138 > xxx.yyy.175.249.138: udp 212 (ttl 127, id 54785)
14:24:19.195835 10.121.201.15.138 > xxx.yyy.147.7.138: udp 212 (ttl 127, id 55041)
14:24:19.833333 10.121.201.15.138 > xxx.yyy.171.248.138: udp 212 (ttl 127, id 56577)
14:24:20.065099 10.121.201.15.138 > xxx.yyy.197.15.138: udp 212 (ttl 127, id 57089)
```

1. Source of trace:

Just outside firewall protecting main corporate subnet of large company.

2. Detect was generated by:

tcpdump running on a UNIX machine just outside the firewall. tcpdump is set to capture all packets (with certain partner network traffic filtered out) and results are passed to Snort and also checked with composite filters based on the ones at the back of the SANS course

guide 2.5, *Intrusion Detection Workshop*, by the GIAC staff.

This particular trace was flagged because the source IP is on net 10 without being on one of the internal 10-nets. The relevant line from the filter is (net 1 or net 2 or net 5 or (net 10 and (not net 10.p and not net 10.q and not net 10.r))), where 10.p, 10.q and 10.r are used internally.

3. Probability that the source address was spoofed:

The unroutable address (this company uses 10. addresses internally, but none near this range) is an indication of a spoofed / crafted address. Then again, it's possible that this attacker's network interface is set to use 10.121.201.15 and he just sort of forgot that nothing will ever be routed back to him.

4. Description of attack:

This is an attempt to find hosts on various subnets of the company's address space that respond to UDP 138, Windows NetBIOS datagram service and possibly to extract information from these servers at the same time. Note that the first series of UDP packets contain 225 bytes of user data, which may well contain NetBIOS data transfer requests. The second series (after a nine-second delay) contain 212 bytes of user data. Possibly the attacker was trying one version of an attack, then paused to switch to the next version.

5. Attack mechanism:

The attacker was probably hoping to build to gather information like server and workgroup name, names of shares, login names of account holders and admin account information using a tool like smbclient (http://www.samba.org). (Testing with smbclient, specifically, on my own network fails to produce traffic resembling the above UDP queries, but that is the basic idea.)

How the attacker expects to retrieve information from this scan with an unroutable address is beyond me.

6. Correlations:

This attack is well known and was discussed in the SANS Intrusion Detection classes. It is a variation on IDS177 as submitted and described by Max Vision. (He described a transfer attack against port 137, but this seems to be pretty well the same thing against port 138.)

7. Evidence of active targeting:

The attacker is probing several hosts, seemingly at random, from a spoofed, or at least invalid, IP address. None of the target IP addresses map to Windows hosts, and some do not map to any hosts at all.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	5	Many critical application and file servers exist on the internal networks being attacked.
Lethality	4	If successful, a significant amount of information is available about the shares and accounts on a given machine, and possibly even data from the hosts.
Countermeasures	2	The security posture of the Windows hosts involved is unknown, but not generally regarded as high.
Network CM	5	Strong firewall protecting internal network.
TOTAL SEVERITY	2	(5 + 4) - (2 + 5)

9. Defensive recommendation:

A review of firewall rules showed that port 138 traffic is not allowed through this gateway, so no new defenses are needed there. Some 138 traffic is allowed through various partner firewalls, but they are VPN's and considered fairly secure. Nevertheless, the appropriate admins were alerted that this was going on and advised to review the security of their hosts.

10. Multiple choice test question:

Given the following trace:

```
14:23:55.188920 10.121.201.15.138 > xxx.yyy.197.148.138: udp 225 (ttl 127, id 22529)
14:23:55.458853 10.121.201.15.138 > xxx.yyy.198.3.138: udp 225 (ttl 127, id 23041)
```

```
14:23:56.628800 10.121.201.15.138 > xxx.yyy.175.249.138: udp 225 (ttl 127, id 25601)
14:23:56.848799 10.121.201.15.138 > xxx.yyy.147.7.138: udp 225 (ttl 127, id 26113)
```

Which statement best characterizes this attack?

a) It is directed primarily at Windows machines.

b) It is directed primarily at Solaris machines.

c) It is directed primarily at network infrastructure devices like routers.

d) It is purely a denial of service attack against a range of machine types.

Answer: a)

Detect 4

```
04:01:17.136768 aaa.bbb.124.50 > xxx.yyy.17.0: icmp: echo reply (ttl 113, id 35181)
04:01:55.684792 aaa.bbb.124.50 > xxx.yyy.101.0: icmp: echo reply (ttl 113, id 60305)
04:02:07.239004 aaa.bbb.124.50 > xxx.yyy.38.0: icmp: echo reply (ttl 113, id 47337)
04:02:09.218525 aaa.bbb.124.50 > xxx.yyy.45.0: icmp: echo reply (ttl 113, id 58615)
04:03:11.984221 aaa.bbb.124.50 > xxx.yyy.7.0: icmp: echo reply (ttl 113, id 9469)
04:03:44.235224 aaa.bbb.124.50 > xxx.yyy.34.0: icmp: echo reply (ttl 113, id 26353)
04:03:44.904892 aaa.bbb.124.50 > xxx.yyy.183.0: icmp: echo reply (ttl 113, id 13814)
04:34:49.895228 aaa.bbb.124.50 > xxx.yyy.193.0: icmp: echo reply (ttl 113, id 26056)
04:35:24.122218 aaa.bbb.124.50 > xxx.yyy.242.0: icmp: echo reply (ttl 113, id 41838)
04:35:39.290671 aaa.bbb.124.50 > xxx.yyy.95.0: icmp: echo reply (ttl 113, id 44555)
04:35:45.904888 aaa.bbb.124.50 > xxx.yyy.1.0: icmp: echo reply (ttl 113, id 49598)
04:35:48.762123 aaa.bbb.124.50 > xxx.yyy.180.0: icmp: echo reply (ttl 113, id 25358)
04:35:49.925918 aaa.bbb.124.50 > xxx.yyy.105.0: icmp: echo reply (ttl 113, id 53807)
04:36:07.044428 aaa.bbb.124.50 > xxx.yyy.64.0: icmp: echo reply (ttl 113, id 43781)
04:36:10.916867 aaa.bbb.124.50 > xxx.yyy.168.0: icmp: echo reply (ttl 113, id 21356)
04:36:15.211373 aaa.bbb.124.50 > xxx.yyy.147.0: icmp: echo reply (ttl 113, id 13282)
04:36:16.354845 aaa.bbb.124.50 > xxx.yyy.129.0: icmp: echo reply (ttl 113, id 18179)
04:36:20.868184 aaa.bbb.124.50 > xxx.yyy.38.0: icmp: echo reply (ttl 113, id 1408)
04:36:31.349822 aaa.bbb.124.50 > xxx.yyy.253.0: icmp: echo reply (ttl 113, id 60064)
04:36:32.175728 aaa.bbb.124.50 > xxx.yyy.29.0: icmp: echo reply (ttl 113, id 44471)
04:36:52.262411 aaa.bbb.124.50 > xxx.yyy.8.0: icmp: echo reply (ttl 113, id 7390)
04:37:00.903765 aaa.bbb.124.50 > xxx.yyy.160.0: icmp: echo reply (ttl 113, id 35531)
04:37:05.028912 aaa.bbb.124.50 > xxx.yyy.162.0: icmp: echo reply (ttl 113, id 19772)
04:37:10.611686 aaa.bbb.124.50 > xxx.yyy.20.0: icmp: echo reply (ttl 113, id 28375)
04:37:19.613608 aaa.bbb.124.50 > xxx.yyy.158.0: icmp: echo reply (ttl 113, id 58573)
04:37:36.776446 aaa.bbb.124.50 > xxx.yyy.206.0: icmp: echo reply (ttl 113, id 56484)
04:37:43.511190 aaa.bbb.124.50 > xxx.yyy.24.0: icmp: echo reply (ttl 113, id 11869)
04:37:50.664722 aaa.bbb.124.50 > xxx.yyy.147.0: icmp: echo reply (ttl 113, id 51234)
04:37:59.969055 aaa.bbb.124.50 > xxx.yyy.245.0: icmp: echo reply (ttl 113, id 46881)
04:38:17.270407 aaa.bbb.124.50 > xxx.yyy.236.0: icmp: echo reply (ttl 113, id 27900)
04:38:24.665858 aaa.bbb.124.50 > xxx.yyy.153.0: icmp: echo reply (ttl 113, id 51654)
04:38:28.599318 aaa.bbb.124.50 > xxx.yyy.242.0: icmp: echo reply (ttl 113, id 10550)
04:38:53.164635 aaa.bbb.124.50 > xxx.yyy.134.0: icmp: echo reply (ttl 113, id 50135)
04:38:54.765081 aaa.bbb.124.50 > xxx.yyy.124.0: icmp: echo reply (ttl 113, id 30723)
04:38:59.884264 aaa.bbb.124.50 > xxx.yyy.79.0: icmp: echo reply (ttl 113, id 37776)
04:39:11.155801 aaa.bbb.124.50 > xxx.yyy.244.0: icmp: echo reply (ttl 113, id 25797)
04:39:19.689428 aaa.bbb.124.50 > xxx.yyy.227.0: icmp: echo reply (ttl 113, id 8880)
04:39:24.633318 aaa.bbb.124.50 > xxx.yyy.121.0: icmp: echo reply (ttl 113, id 58679)
04:39:36.650999 aaa.bbb.124.50 > xxx.yyy.134.0: icmp: echo reply (ttl 113, id 21888)
04:39:53.650253 aaa.bbb.124.50 > xxx.yyy.118.0: icmp: echo reply (ttl 113, id 59474)
04:39:56.714088 aaa.bbb.124.50 > xxx.yyy.117.0: icmp: echo reply (ttl 113, id 8874)
04:40:11.129305 aaa.bbb.124.50 > xxx.yyy.196.0: icmp: echo reply (ttl 113, id 50487)
04:40:13.369510 aaa.bbb.124.50 > xxx.yyy.130.0: icmp: echo reply (ttl 113, id 34417)
04:40:33.087323 aaa.bbb.124.50 > xxx.yyy.193.0: icmp: echo reply (ttl 113, id 63375)
04:40:40.199486 aaa.bbb.124.50 > xxx.yyy.37.0: icmp: echo reply (ttl 113, id 10837)
04:40:46.277812 aaa.bbb.124.50 > xxx.yyy.177.0: icmp: echo reply (ttl 113, id 55027)
04:40:53.764836 aaa.bbb.124.50 > xxx.yyy.72.0: icmp: echo reply (ttl 113, id 9152)
04:40:56.061872 aaa.bbb.124.50 > xxx.yyy.40.0: icmp: echo reply (ttl 113, id 19458)
```

[Both addresses have been sanitized because the source at aaa.bbb.124.50 is innocent.]

1. Source of trace:

Just outside firewall protecting main corporate subnet of large company.

2. Detect was generated by:

tcpdump running on a UNIX machine just outside the firewall. tcpdump is set to capture all packets (with certain partner network traffic filtered out) and results are passed to Snort and also checked with composite filters based on the ones at the back of the SANS course guide 2.5, *Intrusion Detection Workshop*, by the GIAC staff.

This particular trace was flagged because the filter finds .0 addresses. Obviously, all of these packets were flagged, since the destination IP addresses all end in zero. A rerun of tcpdump with host aaa.bbb.124.50 revealed that indeed these are all the packets associated with that host.

3. Probability that the source address was spoofed:

There's spoofing going on, but not against us (directly). Our IP address range is being used to spoof someone else. See below.

aaa.bbb.124.50 is most likely a Windows NT4 / 95 / 98 machine, according to nmap. I assess it as an NT4 machine based on the received TTL value of 113. Win 95 and 98 output packets, by default, with TTL values of 32, and NT4 uses a default value of 128. That is a reasonable starting value based on a received value of 113. This would indicate 15 hops from that machine to my network; a traceroute back to aaa.bbb.124.50 yielded 14 hops, so this seems plausible.

Its netblock is assigned to an educational institution in Hong Kong. Furthermore, the subdomain on which the machine is located (according to DNS) is security.___.hk. (Why would anyone call a collection of security hosts by that name?) The ARIN-listed site coordinator has been e-mailed about the attack, but no response has been received as yet.

4. Description of attack:

We are seeing a series of ICMP echo replies directed at non-existent addresses within my netblock, possibly intending that .0 will respond as if it were a broadcast address. While it is certainly possible that aaa.bbb.124.50 is generating bogus "echo reply" messages and scattering them out to random .0 addresses in and near my netblock, it is much more likely that they were generated in response to ICMP echo request messages received from a third party.

5. Attack mechanism:

Probably some third party is crafting ICMP echo request messages ("pings") with bogus source addresses, many of which happen to be in my netblock. Thus, we see the reply messages at my network, but never see any of the original ping messages. The victim host (aaa.bb.124.50) is probably being attacked with a ping flood, or possibly a ping o'death attack.

I investigated the possibility that this is a Smurf attack, in which the attacker directs pings at a broadcast address (in this case, perhaps aaa.bbb.124.255), and that .50 happens to be the only host responding, either because it's the only host that sees the pings for some reason (filtering, perhaps?) or maybe it's the only live host on that net. This does not appear to be the case, as quite a few machines on that net respond to pings (as confirmed by 'nmap -sP aaa.bbb.124.*').

It doesn't have anything directly to do with the traffic observed in this detect, but nmap reports that the victim host is running a service on port 31337. This machine has very likely been infected with BackOrifice! Someone has been beating on this poor machine hard.

6. Correlations:

Ping flood, ping o'death and Smurf are all well known attacks and the secondary indications of spoofed IP addresses in this case were addressed in the SANS course material in the San José Intrusion Detection course in May, 2000. They are also described well in many sources; a good overview exists at http://www.cisco.com/warp/public/707/22.html.

I compiled a version of ping o'death I found on the web and tried it on my own network. It didn't crash any Solaris, HP-UX or Windows NT machines, but it did cause Solaris machines to respond with about thirty echo replies for every crafted echo request they received.

7. Evidence of active targeting:

Many ping replies in a short period of time combined with the random nature of the IP addresses to which they reply, plus the fact that all target IP addresses end in .0 are indicators of active targeting. The fact that the target machine, judging from the name of its subdomain, is a security host, and that it may have been infected with BackOrifice, are correlating factors.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	1	Hey, it's not my machine he's after.
Lethality	3	The rate of ping replies my network is seeing is not particularly extreme and poses no real threat, either to hosts or bandwidth.
Countermeasures	4	No machines on this subnet should be vulnerable to ping floods or ping o'death.
Network CM	5	Inbound ICMP echo requests and replies are stopped at the firewall at this gateway, with the exception of a few specific hosts with a need to be able to ping through the firewall.
TOTAL SEVERITY	-5	(1 + 3) - (4 + 5)

9. Defensive recommendation:

Make sure we aren't vulnerable to this sort of thing: scan for BackOrifice and related trojans, continue to block ICMP traffic (particularly for .255 and .0 addresses) at the firewall.

10. Multiple choice test question:

Given the following trace:

```
04:01:17.136768 aaa.bbb.124.50 > xxx.yyy.17.0: icmp: echo reply (ttl 113, id 35181)
04:01:55.684792 aaa.bbb.124.50 > xxx.yyy.101.0: icmp: echo reply (ttl 113, id 60305)
04:02:07.239004 aaa.bbb.124.50 > xxx.yyy.38.0: icmp: echo reply (ttl 113, id 47337)
04:02:09.218525 aaa.bbb.124.50 > xxx.yyy.45.0: icmp: echo reply (ttl 113, id 58615)
04:03:11.984221 aaa.bbb.124.50 > xxx.yyy.7.0: icmp: echo reply (ttl 113, id 9469)
```

Indications that spoofing is going on include:

a) ICMP echo replies are arriving when no echo request packets were sent out.

b) Destination hosts all end in .0.

c) All ID fields are odd-numbered.

d) All of the above.

Answer: a)

Detect 5

```
15:26:34.041436 147.32.140.10.3960 > xxx.yyy.56.0.111: S 6848000:6848000(0) win 24576 (ttl 45, id 28269)
15:26:39.793375 147.32.140.10.3960 > xxx.yyy.56.0.111: S 6848000:6848000(0) win 24576 (ttl 45, id 28501)
15:26:51.785313 147.32.140.10.3960 > xxx.yyy.56.0.111: S 6848000:6848000(0) win 24576 (ttl 45, id 28697)
15:28:42.714014 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 30606)
15:28:48.271399 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 30716)
15:29:00.257861 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 31008)
15:29:24.290106 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 31378)
15:29:57.382151 147.32.140.10.3964 > xxx.yyy.56.2.111: S 34112000:34112000(0) win 24576 (ttl 45, id 31635)
15:30:03.320696 147.32.140.10.3964 > xxx.yyy.56.2.111: S 34112000:34112000(0) win 24576 (ttl 45, id 31679)
15:30:15.278608 147.32.140.10.3964 > xxx.yyy.56.2.111: S 34112000:34112000(0) win 24576 (ttl 45, id 31839)
15:30:39.307573 147.32.140.10.3964 > xxx.yyy.56.2.111: S 34112000:34112000(0) win 24576 (ttl 45, id 32057)
15:31:18.303648 147.32.140.10.3968 > xxx.yyy.56.3.111: S 44352000:44352000(0) win 24576 (ttl 45, id 32328)
15:31:54.299226 147.32.140.10.3968 > xxx.yyy.56.3.111: S 44352000:44352000(0) win 24576 (ttl 45, id 32566)
15:32:33.304579 147.32.140.10.3972 > xxx.yyy.56.4.111: S 54592000:54592000(0) win 24576 (ttl 45, id 33392)
15:32:45.314996 147.32.140.10.3972 > xxx.yyy.56.4.111: S 54592000:54592000(0) win 24576 (ttl 45, id 33817)
15:33:42.357010 147.32.140.10.3973 > xxx.yyy.56.5.111: S 64576000:64576000(0) win 24576 (ttl 45, id 35088)
15:33:48.319727 147.32.140.10.3973 > xxx.yyy.56.5.111: S 64576000:64576000(0) win 24576 (ttl 45, id 35189)
15:34:00.339683 147.32.140.10.3973 > xxx.yyy.56.5.111: S 64576000:64576000(0) win 24576 (ttl 45, id 35496)
15:34:24.327830 147.32.140.10.3973 > xxx.yyy.56.5.111: S 64576000:64576000(0) win 24576 (ttl 45, id 35668)
15:34:57.386609 147.32.140.10.3974 > xxx.yyy.56.6.111: S 74432000:74432000(0) win 24576 (ttl 45, id 35891)
15:35:03.330674 147.32.140.10.3974 > xxx.yyy.56.6.111: S 74432000:74432000(0) win 24576 (ttl 45, id 35937)
15:35:15.324600 147.32.140.10.3974 > xxx.yyy.56.6.111: S 74432000:74432000(0) win 24576 (ttl 45, id 36299)
15:36:12.400770 147.32.140.10.3975 > xxx.yyy.56.7.111: S 84416000:84416000(0) win 24576 (ttl 45, id 37062)
15:36:18.343228 147.32.140.10.3975 > xxx.yyy.56.7.111: S 84416000:84416000(0) win 24576 (ttl 45, id 37146)
15:36:30.332153 147.32.140.10.3975 > xxx.yyy.56.7.111: S 84416000:84416000(0) win 24576 (ttl 45, id 37296)
15:36:54.359632 147.32.140.10.3975 > xxx.yyy.56.7.111: S 84416000:84416000(0) win 24576 (ttl 45, id 37570)
15:37:27.477007 147.32.140.10.3979 > xxx.yyy.56.8.111: S 94784000:94784000(0) win 24576 (ttl 45, id 38979)
```

```
15:37:33.350784 147.32.140.10.3979 > xxx.yyy.56.8.111: S 94784000:94784000(0) win 24576 (ttl 45, id 39040)
15:37:45.362957 147.32.140.10.3979 > xxx.yyy.56.8.111: S 94784000:94784000(0) win 24576 (ttl 45, id 39305)
15:38:09.339322 147.32.140.10.3979 > xxx.yyy.56.8.111: S 94784000:94784000(0) win 24576 (ttl 45, id 39616)
15:38:48.374998 147.32.140.10.3981 > xxx.yyy.56.9.111: S 104896000:104896000(0) win 24576 (ttl 45, id 40298)
15:39:00.355345 147.32.140.10.3981 > xxx.yyy.56.9.111: S 104896000:104896000(0) win 24576 (ttl 45, id 40761)
15:39:24.361893 147.32.140.10.3981 > xxx.yyy.56.9.111: S 104896000:104896000(0) win 24576 (ttl 45, id 41382)
15:40:03.374224 147.32.140.10.3982 > xxx.yyy.56.10.111: S 114880000:114880000(0) win 24576 (ttl 45, id 41573)
15:40:15.382377 147.32.140.10.3982 > xxx.yyy.56.10.111: S 114880000:114880000(0) win 24576 (ttl 45, id 41578)
15:40:39.380883 147.32.140.10.3982 > xxx.yyy.56.10.111: S 114880000:114880000(0) win 24576 (ttl 45, id 41683)
15:41:12.481868 147.32.140.10.3987 > xxx.yyy.56.11.111: S 125248000:125248000(0) win 24576 (ttl 45, id 42430)
15:41:18.383990 147.32.140.10.3987 > xxx.yyy.56.11.111: S 125248000:125248000(0) win 24576 (ttl 45, id 42436)
```

[... and much more of the same, right up the whole subnet]

1. Source of trace:

Just outside firewall protecting main corporate subnet of large company.

2. Detect was generated by:

tcpdump running on a UNIX machine just outside the firewall. tcpdump is set to capture all packets (with certain partner network traffic filtered out) and results are passed to Snort and also checked with composite filters based on the ones at the back of the SANS course guide 2.5, *Intrusion Detection Workshop*, by the GIAC staff.

This particular trace was flagged because it attempted to scan xxx.yyy.56.0 and the filter caught it. A rescan of the tcpdump file on the offending host yielded the above.

147.32.140.10 maps to the netblock of the Prague Institute of Chemical Technology, in the Czech Republic. Mail to the ARIN-listed coordinator bounced and no further attempt to contact them has been made. A few attempts were made to nmap this host and find something out about it, but it was not responding by the time I tried.

3. Probability that the source address was spoofed:

This attack wouldn't be much use if the source address were spoofed, as the reply packets wouldn't get back to the source.

4. Description of attack:

This is a scan for rpcbind services, which can give quite a lot of information about RPC services, and therefore the applications, running on a given machine. On most UNIX variants, this service is enabled by default. Each host got a SYN packet directed at port 111 several times over the course of 30 seconds or so. This indicates that the scanning application used a standard system connect(3) library call, as this is normal behavior for connection request timeout. This is correlated by the fact that the packets do not generally appear to be crafted -- for example, the initial sequence numbers increase (sort of) randomly, rather than include a hard-coded ISN.

This isn't the "classic" portmapper attack: classic portmapper scans (such as the one related at http://www.sans.org/y2k/032200-1730.htm, and those discussed in class) generally probe both UDP and TCP port 111, as well as (sometimes) ports above 32770.

The TTL value of 45 sounds as if it may have come from a host with a default initial TTL value of 64 -- other hosts in the .cz domain that were tried with traceroute were 15-20 hops away, which correlates well with this assumption. The most likely platforms fitting this assumption are Linux and FreeBSD.

The attacker happens to have picked a subnet (xxx.yyy.56) that hosts several important Solaris servers. Does he know something about our network already?

5. Attack mechanism:

Once a connection is made on port 111, a simple request can be made that will download detailed information about all registered RPC services, such as NIS, NFS, NFS automounting and many others. This information could then be used to craft attacks that enable an unauthorized person to gain privileged access, or to read data, or to deny service on that host. The UNIX command rpcinfo(1M) makes use of this service, for example.

6. Correlations:

Attacks against port 111 are well known and were discussed in the SANS course material in the San José Intrusion Detection course in May, 2000, though this one does not exactly match those. A thorough description of portmapper and attacks against port 111 in general may be found at http://www.sans.org/newlook/resources/IDFAQ/blocking.htm.

7. Evidence of active targeting:

The sequential stepping over every host in a particular subnet (and one that just happens to have vulnerable hosts within) is a strong indication of active targeting.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	5	Some machines on the targeted subnet are critical application servers.
Lethality	4	If successful, a significant amount of information is available about the services running on these hosts.
Countermeasures	2	All of the Solaris machines involved, I am sad to report, do provide this service. On the other hand, at least their patch levels are pretty current.
Network CM	5	Strong firewall protecting internal network. No connections originating from outside are allowed, and no traffic whatsoever to port 111 of internal hosts is permitted.
TOTAL SEVERITY	3	(5 + 4) - (1 + 5)

9. Defensive recommendation:

Verify that firewall rules do not permit this sort of thing, and shut off the sunrpc service on port 111 of the Solaris machines, unless strong justification exists for retaining it. Keep UNIX machine patches up to date.

10. Multiple choice test question:

Given the following trace:

```
15:26:34.041436 147.32.140.10.3960 > xxx.yyy.56.0.111: S 6848000:6848000(0) win 24576 (ttl 45, id 28269)
15:26:39.793375 147.32.140.10.3960 > xxx.yyy.56.0.111: S 6848000:6848000(0) win 24576 (ttl 45, id 28501)
15:26:51.785313 147.32.140.10.3960 > xxx.yyy.56.0.111: S 6848000:6848000(0) win 24576 (ttl 45, id 28697)
15:28:42.714014 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 30606)
15:28:48.271399 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 30716)
15:29:00.257861 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 31008)
15:29:24.290106 147.32.140.10.3963 > xxx.yyy.56.1.111: S 24128000:24128000(0) win 24576 (ttl 45, id 31378)
```

What attack is this?

a) SYN Flood

b) Probe for RPC service information

c) Land attack

d) Third-party reset scan of 147.32.140.10 -- we are merely seeing the resulting SYN packets

Answer: b)

Detect 6

```
05:49:14.365488 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 51911)
05:49:15.362663 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 51957)
05:49:25.351813 204.253.131.35.2000 > xxx.yyy.240.0.23: S 67112:67112(0) win 8760 (DF) (ttl 115, id 14371)
05:49:30.355620 204.253.131.35.2000 > xxx.yyy.240.0.21: S 67127:67127(0) win 8760 (DF) (ttl 115, id 14388)
05:49:35.351518 204.253.131.35.2000 > xxx.yyy.240.0.80: S 67140:67140(0) win 8760 (DF) (ttl 115, id 14404)
06:04:14.378774 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 57760)
06:04:15.378116 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 57806)
06:04:25.362002 204.253.131.35.2000 > xxx.yyy.240.0.23: S 68191:68191(0) win 8760 (DF) (ttl 115, id 15644)
06:04:30.364914 204.253.131.35.2000 > xxx.yyy.240.0.21: S 68206:68206(0) win 8760 (DF) (ttl 115, id 15661)
06:04:35.362148 204.253.131.35.2000 > xxx.yyy.240.0.80: S 68219:68219(0) win 8760 (DF) (ttl 115, id 15677)
06:19:29.365363 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 63658)
06:19:40.367760 204.253.131.35.2000 > xxx.yyy.240.0.23: S 69284:69284(0) win 8760 (DF) (ttl 115, id 16936)
06:19:45.367860 204.253.131.35.2000 > xxx.yyy.240.0.21: S 69300:69300(0) win 8760 (DF) (ttl 115, id 16952)
06:19:50.365443 204.253.131.35.2000 > xxx.yyy.240.0.80: S 69315:69315(0) win 8760 (DF) (ttl 115, id 16971)
```

[This went on like this, the same thing every 15 minutes, for over two full days.]

1. Source of trace:

Just outside firewall protecting main corporate subnet of large company.

2. Detect was generated by:

tcpdump running on a UNIX machine just outside the firewall. tcpdump is set to capture all packets (with certain partner network traffic filtered out) and results are passed to Snort and also checked with composite filters based on the ones at the back of the SANS course guide 2.5, *Intrusion Detection Workshop*, by the GIAC staff.

This particular trace was flagged because the destination IP ends in .0 and the filter flags all such packets. A recheck on the host 204.253.131.35 showed that this was the extent of the traffic.

3. Probability that the source address was spoofed:

Slightly possible. This might be some kind of new attack where the attacker is hoping .0 addresses will act as broadcast addresses, causing hundreds of SYN-ACK and/or RESET-ACK packets to go flooding back to poor 204.253.131.35. (Sort of a TCP version of Smurf?) Same for the ping (icmp echo request) packets: lots of echo replies would go flooding back if this worked.

Another possibility is that this attacker is truly at 204.253.131.35 and is probing for services on well-known ports 23, 21 and 80 (telnet, ftp and http, respectively), hoping to find ways to compromise a system. The source port is always 2000, which is odd, and an indication of crafted packets.

nmap didn't have a good guess about the platform, but found that it was providing services on ports 21 (ftp), 22 (ssh), 23 (telnet), 80 (http) and 2000.

A test with an ssh client confirmed that the host is indeed running ssh on port 22. A test with ftp showed that the host is running something called ArrowPoint 5.3.1, which seems to be a commercial ftp/http load balancer. Only empty documents could be found on HTTP port 80, though a web service was running. (www.netcraft.com was unable to determine what server software and OS, though). Port 2000 is, according to IANA, a port for "callbook" services, an amateur radio directory of callsigns. This is not the service running at port 2000 on this machine, though.

The machine's IP address is in a netblock belonging to UUNET, but no DNS information was available regarding it. Numerous domains (all apparently small businesses) share pieces of the class C subnet on which this host resides.

In short, I used every trick I have and was unable to identify this machine with any degree of reliability! However, the fact that it runs a service on port 2000 and it is using port 2000 in its connection attempts is too much of a coincidence to ignore. I conclude that the source address is not spoofed.

4. Description of attack:

Every fifteen minutes, this host sends a TCP SYN packet (containing no data) to ports 23, 21 and 80 of xxx.yyy.240.0, which isn't even a valid host address. It seems reasonable to assume that my net is not the only one being scanned in this way, and that I am seeing only "slices" of a scan that goes through perhaps hundreds or thousands of addresses, then repeats. For two days. What is this guy thinking?

5. Attack mechanism:

As hinted in the last comment, I am fairly mystified as to what this attacker hopes to accomplish. If he finds services running (by getting a SYN-ACK) on TCP ports 21, 23 and/or 80, that information can be used to design attacks against hosts later.

6. Correlations:

I was unable to find previously catalogued attacks resembling this one. Nothing like this was discussed in class.

7. Evidence of active targeting:

The regularity and span of time over which this odd-looking connection attempt occurred, and the unwavering source port of 2000, indicates that this is an active attack.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	5	This network contains important file and application servers, and a firewall.
Lethality	3	Though it's not clear what the attacker is after, some information about running services could be obtained if he should get lucky and hit a host running services on TCP 21, 23 or 80.
Countermeasures	2	Many hosts inside the network serve ftp and telnet on ports 21 and 23, respectively. There are a few internal web servers serving TCP port 80, as well. The latter are all NT4 machines of unknown security posture.
Network CM	5	Strong firewall protecting internal network. No connections originating from outside are allowed.
TOTAL SEVERITY	1	(5 + 3) - (2 + 5)

9. Defensive recommendation:

Nothing really to be done; the firewall is stopping this traffic by forbidding connections initiated from outside the firewall to internal hosts.

10. Multiple choice test question:

Given the following trace, repeated every 15 minutes for two full days:

```
05:49:14.365488 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 51911)
05:49:15.362663 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 51957)
05:49:25.351813 204.253.131.35.2000 > xxx.yyy.240.0.23: S 67112:67112(0) win 8760 (DF) (ttl 115, id 14371)
05:49:30.355620 204.253.131.35.2000 > xxx.yyy.240.0.21: S 67127:67127(0) win 8760 (DF) (ttl 115, id 14388)
05:49:35.351518 204.253.131.35.2000 > xxx.yyy.240.0.80: S 67140:67140(0) win 8760 (DF) (ttl 115, id 14404)
06:04:14.378774 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 57760)
06:04:15.378116 204.253.131.35 > xxx.yyy.240.0: icmp: echo request (ttl 51, id 57806)
06:04:25.362002 204.253.131.35.2000 > xxx.yyy.240.0.23: S 68191:68191(0) win 8760 (DF) (ttl 115, id 15644)
06:04:30.364914 204.253.131.35.2000 > xxx.yyy.240.0.21: S 68206:68206(0) win 8760 (DF) (ttl 115, id 15661)
06:04:35.362148 204.253.131.35.2000 > xxx.yyy.240.0.80: S 68219:68219(0) win 8760 (DF) (ttl 115, id 15677)
```

What is the objective of this attack?

a) Search for trojans on Windows machines

b) Search for trojans on UNIX machines

c) Denial of service against telnet, ftp and web services

d) ~~I don't have a bloody clue~~ None of the above

Answer: d)

Detect 7

```
11:41:14.439059 aaa.bbb.239.218.51551 > xxx.yyy.151.40.1651: R 0:0(0) ack 674719802 win 0 (ttl 51, id 49872)
11:48:11.758100 aaa.bbb.239.218.49043 > xxx.yyy.169.234.2110: R 0:0(0) ack 674719802 win 0 (ttl 51, id 51770)
11:52:47.049856 aaa.bbb.239.218.38238 > xxx.yyy.139.145.1155: R 0:0(0) ack 674719802 win 0 (ttl 51, id 52066)
11:58:18.135597 aaa.bbb.239.218.63233 > xxx.yyy.44.27.1457: R 0:0(0) ack 674719802 win 0 (ttl 51, id 5789)
12:11:54.874933 aaa.bbb.239.218.46719 > xxx.yyy.72.127.2398: R 0:0(0) ack 674719802 win 0 (ttl 51, id 54347)
12:24:21.731061 aaa.bbb.239.218.50218 > xxx.yyy.235.121.1131: R 0:0(0) ack 674719802 win 0 (ttl 51, id 43860)
12:38:25.683607 aaa.bbb.239.218.5737 > xxx.yyy.18.110.1950: R 0:0(0) ack 674719802 win 0 (ttl 51, id 11448)
13:21:01.567670 aaa.bbb.239.218.13628 > xxx.yyy.116.173.2111: R 0:0(0) ack 674719802 win 0 (ttl 51, id 61921)
13:24:13.594448 aaa.bbb.239.218.13561 > xxx.yyy.6.255.1348: R 0:0(0) ack 674719802 win 0 (ttl 51, id 58372)
13:26:59.454192 aaa.bbb.239.218.63341 > xxx.yyy.2.79.1153: R 0:0(0) ack 674719802 win 0 (ttl 51, id 57524)
13:29:06.056107 aaa.bbb.239.218.19793 > xxx.yyy.202.142.1790: R 0:0(0) ack 674719802 win 0 (ttl 51, id 23197)
13:46:46.886528 aaa.bbb.239.218.14978 > xxx.yyy.239.4.1126: R 0:0(0) ack 674719802 win 0 (ttl 51, id 12518)
13:52:28.145279 aaa.bbb.239.218.34894 > xxx.yyy.95.144.2039: R 0:0(0) ack 674719802 win 0 (ttl 51, id 33728)
13:57:03.943021 aaa.bbb.239.218.12884 > xxx.yyy.118.60.2322: R 0:0(0) ack 674719802 win 0 (ttl 51, id 35600)
```

[Both addresses have been sanitized because the source at aaa.bbb.239.218 is innocent.]

[No packets have been deleted -- this is all that was captured from this host in the 48-hour period examined.]

1. Source of trace:

Just outside firewall protecting main corporate subnet of large company.

2. Detect was generated by:

tcpdump running on a UNIX machine just outside the firewall. tcpdump is set to capture all packets (with certain partner network traffic filtered out) and results are passed to Snort and also checked with composite filters based on the ones at the back of the SANS course guide 2.5, *Intrusion Detection Workshop*, by the GIAC staff.

This particular trace was flagged because the filter finds broadcast addresses -- when the packet at 13:24:13 came up (destination address ...255), further investigation (filtering on the source host) revealed the rest of the attack.

3. Probability that the source address was spoofed:

There's spoofing going on, but not against us (directly). Our IP addresses are being used to spoof someone else. See below.

4. Description of attack:

Someone is using addresses in our network range (and probably a lot of others as well) to spoof in an attack against various ports on aaa.bbb.239.218, a host that responds to pings but is not, apparently, a web or mail server. The netblock containing this host is an ISP on the east coast of the US.

All of the packets are RESET-ACK packets, indicating a response from the target host (aaa.bb.239.218, that is) on a port not in use. But no packets originated from these hosts. In fact, most of these hosts do not exist. This indicates that there is a missing piece.

The missing piece is probably a series of SYN packets sent by some third party with xxx.yyy.---.--- addresses in the "source" field of the IP header, directed against various TCP ports on aaa.bbb.239.218. When the targeted port responds with RESET-ACK to the spoofed address, that packet gets routed to my network. If this is the case, we are seeing the "back side" of a SYN flood attack directed against aaa.bbb.239.218.

It is also possible that this is a RESET scan directed against our network, but I am coming down on the side of SYN flood for the following reasons:

- a RESET scan would be more likely to be directed against ports of interest like 23, 53, etc., instead of the random few ports on the random few hosts we see here
- a RESET scan would probably attempt to get better coverage; the extremely random nature of the host selection (and note that one .255 broadcast address is mixed in with the other hosts) indicates that our network is not the one of interest to this attacker
- the very slow and random (several minutes between packets) nature of the detects argues in favor of SYN flood (although it is possible this is just extremely "low and slow"). What we are seeing is probably just that "slice" of IP addresses the attacker is using to spoof that get routed back to us. In between these packets are thousands of other spoofed ones that get routed elsewhere.

The acknowledgement number field (674719802) is the same in every packet. This is a big red flag indicating that the packets were crafted. It indicates that the packets were sent with an Initial Sequence Number of 674719801, which is not only odd but turns out to be a known signature element of a hacker tool called synk4.c. (http://packetstorm.securify.com/opensec-exploits/exploits/DoS/synk4.c. I found this with a web search on '0x28376839' which is the hex representation of 674719801.)

I got a copy of synk4.c and compiled and ran it on my local network. The results from tcpdump were similar to the detect shown above, except that the TCP window size was 65535, instead of 0 as above. Perhaps a variant of synk4.c is floating around out there.

For a look at the "front side" of an attack like this, check out the tcpdump reprinted at http://www.cctec.com/maillists/nanog/historical/9710/msg00059.html. This shows the SYN flood itself, with all the spoofed addresses, that might have generated the traffic above on the "back side."

5. Attack mechanism:

Assuming this is a SYN flood against aaa.bbb.239.218, it works by filling the victim host's TCP/IP stack's tables with bogus pending connections -- the server waits for each connection to complete, which never happens. Once the table fills up, new connection requests are ignored. This can have one of two effects:

- The server's TCP/IP stack crashes; or
- Legitimate users cannot connect.

Either way, it's a denial of service.

SYN floods are sometimes used in conjunction with other attacks, in order to prevent a server from responding to a legitimate request,

as with hijacking a TCP connection.

If it is a RESET scan attempt, it works by sending bogus RESET packets to hosts. Packets sent to hosts that exist disappear silently. Packets sent to hosts that do not exist cause the target site's routers to respond with a message to the effect that the host is unreachable. In this way, a map of live hosts at the site can be constructed.

6. Correlations:

A similar scan, including the crafted ack number, was reported as early as September 1998 by Stephen Northcutt, who identified it as a "reset scan" (http://www.nswc.navy.mil/ISSEC/CID/co-ordinated_analysis.txt). However, that differed from the example at hand in that it dealt with attacks against multiple hosts. This one is an attack against multiple ports on the same host.

More recently (October 99, at http://packetstorm.securify.com/papers/IDS/intv2.txt), Richard Bejtlich ID'ed this as the secondary indications of a SYN flood on the part of some third party. He does not ID the tool (he calls it "The Mystery Tool?").

A reference to "synk4/synk5.c" was made by Fredrik Ostergren in a message reprinted in the SANS Detects Analyzed at http://www.sans.org/y2k/010700-1115.htm, but there were no accompanying technical details. I am not aware of any other sources that associate 674719801 specifically with synk4.c.

I considered this association important enough that I submitted this detect, with an abbreviated version of this analysis, to GIAC at handler@incidents.org,and it was published in the detects for June 15th, at http://www.sans.org/y2k/061500.htm.

7. Evidence of active targeting:

The completely random nature of the destination IP addresses and the range of port numbers on both sides, along with the ack sequence number (as a secondary indication of a crafted ISN) are all strong indicators that this is a deliberate attack.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	1	This attack is not directed against my net.
Lethality	3	No real threat is posed by a bunch of RESET-ACK packets, especially at so low a rate.
Countermeasures	4	No machines on this net should be vulnerable to this kind of attack. They are all recently patched and well maintained.
Network CM	5	The firewall stops packets like this (RESET-ACK when there is no recent outbound traffic to that host) destined for the internal net.
TOTAL SEVERITY	-5	(1 + 3) - (4 + 5)

9. Defensive recommendation:

No additional defenses are required at my end; RESET-ACK traffic is already blocked (when no corresponding outbound SYN exists) and this attack isn't against me, in any case. (If it were, my firewall and existing intrusion detection tools should provide a reasonable defense and warning system.)

I obtained the contact number for the netblock containing the host in question from ARIN. I called the contact name listed there and left a message. I also sent e-mail. As of this writing, I have not been contacted back.

10. Multiple choice test question:

Given the following trace:

```
12:11:54.874933 aaa.bbb.239.218.46719 > xxx.yyy.72.127.2398: R 0:0(0) ack 674719802 win 0 (ttl 51, id 54347)
12:24:21.731061 aaa.bbb.239.218.50218 > xxx.yyy.235.121.1131: R 0:0(0) ack 674719802 win 0 (ttl 51, id 43860)
12:38:25.683607 aaa.bbb.239.218.5737 > xxx.yyy.18.110.1950: R 0:0(0) ack 674719802 win 0 (ttl 51, id 11448)
13:21:01.567670 aaa.bbb.239.218.13628 > xxx.yyy.116.173.2111: R 0:0(0) ack 674719802 win 0 (ttl 51, id 61921)
13:24:13.594448 aaa.bbb.239.218.13561 > xxx.yyy.6.255.1348: R 0:0(0) ack 674719802 win 0 (ttl 51, id 58372)
```

What indicates that crafted packets are involved in this attack?

a) Too-low TTL value

b) Impossible packet ID's

c) Identical sequence numbers

d) All of the above

Answer: c)

Detect 8

```
13:17:09.358846 157.25.5.55.44517 > xxx.yyy.240.23.33474:
        udp 12 [ttl 1] (id 10330)
13:17:09.358969 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33474 unreachable (DF) (ttl 255, id 50674)
13:17:09.359168 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33474 unreachable (DF) (ttl 255, id 50674)
13:17:09.561744 157.25.5.55.44517 > xxx.yyy.240.23.33475:
        udp 12 [ttl 1] (id 10340)
13:17:09.561863 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33475 unreachable (DF) (ttl 255, id 50675)
13:17:09.562063 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33475 unreachable (DF) (ttl 255, id 50675)
13:17:09.643272 157.25.5.55.44517 > xxx.yyy.240.23.33476:
        udp 12 [ttl 1] (id 10343)
13:17:09.643354 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33476 unreachable (DF) (ttl 255, id 50676)
13:17:09.643574 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33476 unreachable (DF) (ttl 255, id 50676)
13:17:49.003637 157.25.5.55.44537 > xxx.yyy.240.23.33474:
        udp 12 [ttl 1] (id 11151)
13:17:49.003742 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33474 unreachable (DF) (ttl 255, id 50677)
13:17:49.003960 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33474 unreachable (DF) (ttl 255, id 50677)
13:17:49.098035 157.25.5.55.44537 > xxx.yyy.240.23.33475:
        udp 12 [ttl 1] (id 11156)
13:17:49.098182 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33475 unreachable (DF) (ttl 255, id 50678)
13:17:49.098400 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33475 unreachable (DF) (ttl 255, id 50678)
13:17:49.178993 157.25.5.55.44537 > xxx.yyy.240.23.33476:
        udp 12 [ttl 1] (id 11157)
13:17:49.179086 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33476 unreachable (DF) (ttl 255, id 50679)
13:17:49.179306 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33476 unreachable (DF) (ttl 255, id 50679)
```

1. Source of trace:

Unused (as yet) Internet connection for new data center of a large midwestern company. xxx.yyy.240 is the network of the outside-the-firewall DMZ.

2. Detect was generated by:

tcpdump (no filtering) running on a UNIX machine attached to a switch just inside the Internet router (but outside of the firewall). Traffic on this net is low enough that tcpdump files may be examined by hand once a few authorized hosts are filtered out.

3. Probability that the source address was spoofed:

The traffic itself does not indicate spoofing. However, the source host appears to be a router (nmap fingerprinting pegs it as Cisco IOS v. 11 or 12), which doesn't particularly make sense given the type of attack this probably is. So it might well be spoofed by a host on the other side of that router (or maybe the router is configured to hide traffic that originates from behind it), and the results gathered by examining traffic that returns to the router. 157.25.5.55 is in a netblock belonging to an ISP in Warsaw, Poland.

4. Description of attack:

Someone is using traceroute to find the route to one of the few live hosts (.23) on this net.

Telltale indications of this are the udp port range: when UDP packets in the range 33434-33600 (and only in that range) are seen, incrementing by one with each packet, it usually means traceroute. (See the next section for references and a description of how traceroute works.) The above detect shows two separate traceroute attempts: one at 13:17:09 and another at 13:17:49. They are essentially identical, but for the source port on the attacking host.

Another signature of traceroute is that the packets come in groups of three, with increasing destination ports, which is exactly what we see here.

The final, glaring, indication that this is traceroute at work is the TTL value of 1, followed by multiple ICMP "UDP port _____ unreachable" messages outbound.

5. Attack mechanism:

Traceroute starts by sending three UDP packets to ports 33434, 33435 and 33436 of the target host, with a TTL value of 1. When it gets a "host unreachable" back, it records that message's source and tries again with the next three ports in sequence, and a TTL value of 2. It repeats this until it finally reaches, or gets an ICMP unreachable port message back from, the host it was trying to find.

Backward-engineering the packet numbers, we find that 33476 == 33434 + 3 * 14, so the machine pinging us is 14 hops away. A traceroute back to this host took 19 hops. This is a bit of a discrepancy, but not alarming enough to conclude definite spoofing. It's likely that the packets simply take different routes depending on direction.

6. Correlations:

Traceroute has been around for over a decade and its source is widely available, so its signature is well-known. It and its signature are described thoroughly at http://www.robertgraham.com/pubs/firewall-seen.html. Some additional information on traceroute and its variants (including a stealthier version for circumventing firewalls) is available at http://www.packetfactory.net/Projects/Firewalk/.

7. Evidence of active targeting:

The traceroute signature is rock-solid evidence that this is the tool used and therefore active targeting is a given.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	2	This machine does nothing of any importance. It is being used temporarily as an intrusion machine, but will be converted to another use soon.
Lethality	2	Traceroute, in this situation, shows nothing particularly secret -- only the route to this network and this and any hosts on it.
Countermeasures	4	The host attacked is fairly well locked down, recently patched, with all unnecessary services turned off, administered through ssh, etc.
Network CM	1	Nothing protects this machine from the internet; it is outside the firewall at this gateway.
TOTAL SEVERITY	-1	(2 + 2) - (4 + 1)

9. Defensive recommendation:

There's not a lot to be done; this host cannot be moved inside the firewall given its current and planned use, and therefore its existence cannot be kept secret from probes. Everything important behind this gateway is protected by a robust firewall that does not permit UDP traffic inbound unless an internal host has already established a "conversation" with the sending host.

10. Multiple choice test question:

Given the following trace:

```
13:17:09.358846 157.25.5.55.44517 > xxx.yyy.240.23.33474:
        udp 12 [ttl 1] (id 10330)
13:17:09.358969 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33474 unreachable (DF) (ttl 255, id 50674)
13:17:09.359168 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33474 unreachable (DF) (ttl 255, id 50674)
13:17:09.561744 157.25.5.55.44517 > xxx.yyy.240.23.33475:
        udp 12 [ttl 1] (id 10340)
13:17:09.561863 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
```

```
        udp port 33475 unreachable (DF) (ttl 255, id 50675)
13:17:09.562063 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33475 unreachable (DF) (ttl 255, id 50675)
13:17:09.643272 157.25.5.55.44517 > xxx.yyy.240.23.33476:
        udp 12 [ttl 1] (id 10343)
13:17:09.643354 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33476 unreachable (DF) (ttl 255, id 50676)
13:17:09.643574 xxx.yyy.240.23 > 157.25.5.55: icmp: xxx.yyy.240.23
        udp port 33476 unreachable (DF) (ttl 255, id 50676)
```

What is this?

a) Probe for a trojan on high ports.

b) Probe for Netware Directory Services on high ports.

c) A probe by the traceroute utility.

d) Could be either a) or b).

Answer: c)

Detect 9

```
Jun 3 00:34:01 cc1014244-a kernel: securityalert: tcp if=ef0
   from 24.3.6.190:4421 to 24.3.21.199 on unserved port 27374
Jun 3 00:41:57 cc1014244-a kernel: securityalert: tcp if=ef0
   from 24.3.6.190:2649 to 24.3.21.199 on unserved port 27374
Jun 3 00:57:49 cc1014244-a kernel: securityalert: tcp if=ef0
   from 24.3.6.190:2086 to 24.3.21.199 on unserved port 27374
Jun 3 01:51:17 cc1014244-a kernel: securityalert: tcp if=ef0
   from 24.3.9.15:1567 to 24.3.21.199 on unserved port 27374
```

[seemingly unrelated detects were deleted]

1. Source of trace:

The GIAC detects for June 5th, at http://www.sans.org/y2k/060500.htm. Found by "Binette," an @home user. @home users generally have more money and bandwidth than sense (nothing personal against Binette, who seems to be an exception to the rule!) and are, therefore, a favorite target of hackers.

2. Detect was generated by:

This detect is from a firewall log.

It shows that the firewall denied access to an attempt to connect via TCP to "unserved port" 27374 through interface ef0. The destination IP is 24.3.21.199. The source (attacker) IP address was 24.3.6.190, with source ports 4421, 2649 and 2086.

3. Probability that the source address was spoofed:

There is no reason, from this detect, to think the source address was spoofed.

4. Description of attack:

This is a scan for a trojan port (27374) for a trojan program known as "SubSeven," version 2.1. In some versions of SubSeven it is also accompanied by probes against port 1243, but that was not seen here. SubSeven and its signature are described thoroughly at http://www.sans.org/y2k/subseven.htm.

5. Attack mechanism:

SubSeven is very popular and getting more so. Most trojans are developed and then just sort of thrown out there -- sooner or later, they are found and neutralized by virus-scanning software. SubSeven, however, is sophisticated and still under active development.

SubSeven is a client-server program. The server, server.exe, is installed on the victim machine (perhaps with an e-mail attachment, or

directly by an attacker with physical access) and infects various system files. It may then notify the attacker (e.g., via ICQ) once it is active, and the attacker will pretty much be in control of the entire system (using the client half of the program).

6. Correlations:

SubSeven "trojan pings" were covered briefly in the SANS Intrusion Detection course in San José. Binette appears to have been suffering from this for some time, as an almost identical detect appears in the course guide, with the same target host.

SubSeven is described thoroughly by Georg Wagner at http://www.sans.org/y2k/subseven.htm.

7. Evidence of active targeting:

The repeated probes, over about half an hour, to port 27374, strongly indicate an active search for the trojan.

The later probe, at 01:51:17, may or may not be related to the first. It is possible that it could be the same machine with a newly assigned IP address from DHCP. @home, the domain of both machines, sometimes does this, and both IP addresses are from the COMCAST MD netblock. Note that this is the same netblock that DNS-probed me back in Detect 1. It's a big netblock, and they probably aren't related, but it bears watching.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	2	This is a guess; from context, it seems to be Binette's home machine, and likely contains data whose compromise or deletion would be painful, but probably wouldn't involve hundreds of thousands or millions of dollars.
Lethality	5	If a SubSeven trojan were present, the attacker pretty much has carte blanche on the system.
Countermeasures	4	Another guess, but if Binette is tracking this stuff this closely, it's a safe bet the machine is pretty well locked down.
Network CM	5	Yet another guess, but for the same reasons as above, the firewall is probably pretty robust and well-maintained.
TOTAL SEVERITY	-2	(2 + 5) - (4 + 5)

9. Defensive recommendation:

The situation is fine, the traffic is being blocked as desired. Nevertheless, it would probably be a good idea to check the machine for SubSeven versions as outlined at the URL given above.

10. Multiple choice test question:

Given the following trace:

```
Jun 3 00:34:01 cc1014244-a kernel: securityalert: tcp if=ef0
    from 24.3.6.190:4421 to 24.3.21.199 on unserved port 27374
Jun 3 00:41:57 cc1014244-a kernel: securityalert: tcp if=ef0
    from 24.3.6.190:2649 to 24.3.21.199 on unserved port 27374
```

Which statement is most likely true?

a) This is an attack against most UNIX variants.

b) This is an indication of a DOS attack intended to hang TCP/IP stacks of Windows NT machines.

c) This is a probe for a Windows trojan.

d) This is a probe for back-door web services.

Answer: c)

Detect 10

```
00:23:12.272490 203.177.11.196.1418 > xxx.yyy.0.50.98: S 3154840318:3154840318(0)
        win 32120 <mss 1460,sackOK,timestamp 6428211[|tcp]> (DF) (ttl 49, id 10850)

00:23:12.320171 203.177.11.196.1419 > xxx.yyy.0.51.98: S 3146254879:3146254879(0)
        win 32120 <mss 1460,sackOK,timestamp 6428234[|tcp]> (DF) (ttl 49, id 10853)

00:23:12.327858 203.177.11.196.1420 > xxx.yyy.0.52.98: S 3151667122:3151667122(0)
        win 32120 <mss 1460,sackOK,timestamp 6428235[|tcp]> (DF) (ttl 49, id 10854)

00:23:12.336925 203.177.11.196.1421 > xxx.yyy.0.53.98: S 3147479060:3147479060(0)
        win 32120 <mss 1460,sackOK,timestamp 6428235[|tcp]> (DF) (ttl 49, id 10855)

00:23:12.345286 203.177.11.196.1422 > xxx.yyy.0.54.98: S 3159665039:3159665039(0)
        win 32120 <mss 1460,sackOK,timestamp 6428236[|tcp]> (DF) (ttl 49, id 10857)
```

[18582 more lines of the same deleted]

```
00:27:37.180893 203.177.11.196.3147 > xxx.yyy.255.177.98: S 3426019553:3426019553(0)
        win 32120 <mss 1460,sackOK,timestamp 6454697[|tcp]> (DF) (ttl 49, id 12294)

00:27:37.189100 203.177.11.196.3148 > xxx.yyy.255.178.98: S 3432174459:3432174459(0)
        win 32120 <mss 1460,sackOK,timestamp 6454698[|tcp]> (DF) (ttl 49, id 12333)

00:27:37.360689 203.177.11.196.3151 > xxx.yyy.255.181.98: S 3434998745:3434998745(0)
        win 32120 <mss 1460,sackOK,timestamp 6454698[|tcp]> (DF) (ttl 49, id 12336)

00:27:37.387037 203.177.11.196.3155 > xxx.yyy.255.185.98: S 3429782511:3429782511(0)
        win 32120 <mss 1460,sackOK,timestamp 6454698[|tcp]> (DF) (ttl 49, id 12340)

00:27:45.168091 203.177.11.196.3130 > xxx.yyy.255.160.98: S 3424936954:3424936954(0)
        win 32120 <mss 1460,sackOK,timestamp 6454697[|tcp]> (DF) (ttl 49, id 12277)
```

[at this point, the attack stops for a while, then ...]

```
01:16:49.571301 203.177.11.196.3678 > xxx.yyy.86.199.98: R 3255581269:3255581269(0)
        win 35013 (ttl 61, id 27661)
01:20:18.952961 203.177.11.196.4916 > xxx.yyy.138.69.98: R 3307176323:3307176323(0)
        win 35013 (ttl 61, id 27661)
01:23:37.376322 203.177.11.196.2854 > xxx.yyy.52.118.98: R 3216007452:3216007452(0)
        win 35013 (ttl 61, id 27661)
01:48:33.368631 203.177.11.196.2945 > xxx.yyy.115.0.98: R 3288424158:3288424158(0)
        win 35013 (ttl 61, id 27661)
01:53:25.034176 203.177.11.196.3561 > xxx.yyy.24.36.98: R 3172835856:3172835856(0)
        win 35013 (ttl 61, id 27661)
```

[66 more lines of same deleted]

```
09:28:03.101814 203.177.11.196.3697 > xxx.yyy.55.195.98: R 3223205368:3223205368(0)
        win 35013 (ttl 61, id 27661)
09:30:11.265348 203.177.11.196.4698 > xxx.yyy.106.83.98: R 3284724377:3284724377(0)
        win 35013 (ttl 61, id 27661)
09:34:49.061957 203.177.11.196.3721 > xxx.yyy.102.126.98: R 3274686070:3274686070(0)
        win 35013 (ttl 61, id 27661)
09:44:28.346410 203.177.11.196.2920 > xxx.yyy.68.68.98: R 3244784624:3244784624(0)
        win 35013 (ttl 61, id 27661)
09:44:48.051664 203.177.11.196.2966 > xxx.yyy.99.137.98: R 3270436053:3270436053(0)
        win 35013 (ttl 61, id 27661)
```

[The capture ends here, but the attack may have continued a while longer.]

1. Source of trace:

Just outside firewall protecting main corporate subnet of large company.

2. Detect was generated by:

tcpdump running on a UNIX machine just outside the firewall. tcpdump is set to capture all packets (with certain partner network traffic filtered out) and results are passed to Snort and also checked with composite filters based on the ones at the back of the SANS course guide 2.5, *Intrusion Detection Workshop*, by the GIAC staff.

The portion of the tcpdump filter that initially flagged this was the (ip[19] = 0xff) or (ip[19] = 0x00) line. (The packets that triggered that line are not shown, but represent destination IP addresses ending in 0 or 255.) A scan on the source IP revealed the rest of the traffic.

3. Probability that the source address was spoofed:

Low, given that this is a probing / mapping attempt, and the attack is useless if the information can't get back to the attacker. Still, the packets are crafted, and spoofing is a possibility.

4. Description of attack:

At first, this is a sequential host sweep for port 98, trying to find machines running linuxconf, a web-based configuration tool for Linux and possible back door into a misconfigured Linux system. It is followed by a much slower reset scan directed at machines, also looking for port 98. Actually, the reset scan isn't necessarily slower, we're just seeing it slower. See below for more on this.

The attack originates from a netblock used by an ISP in the Philippines. nmap's best guess is that it is a Linux machine.

5. Attack mechanism:

In the first portion, thousands of SYN packets containing no data (the "S" and "(0)" parts of the tcpdump lines) are sent to IP addresses within my netblock, pretty much sequentially. Note that in the first part of the detect, source port 1418 is used to probe xxx.yyy.0.50, 1419 used against xxx.yyy.0.51, etc. This pattern was more or less maintained throughout the attack. The only source ports used throughout the attack ranged from 1024 to 4999, repeating as necessary.

Every initial sequence number sent is in the range [3145231913, 3439065511], also a rather "unnatural" characteristic for TCP.

The timestamps show that this sequential scan using SYN packets happened very quickly -- thousands of packets in just over four minutes.

Not only xxx.yyy.0 and xxx.yyy.255 hosts were scanned, but every subnet in between. 0 and 255 are the only ones shown because they bookend the attack. Every single class C subnet (0-255) on this netblock had at least some hosts scanned.

The IANA lists TCP (and UDP) port 98 as something called "TAC news" maintained by the late John Postel. Dozens of informal sources confirmed, however, that port 98 is by far most commonly used for linuxconf services. (For example, at http://www.robertgraham.com/pubs/firewall-seen.html.)

The SYN attack works by detecting hosts that run services on TCP port 98 (because they will send back a SYN-ACK). The attacker can then either note the host in his database for attempting to exploit linuxconf later, or attempt to exploit it immediately. There are several attacks against linuxconf, including buffer overrun attacks and simple misconfigurations that allow control of the host with either no password, or an easily guessable one.

Once the SYN packets end, nearly an hour goes by and then we begin seeing much more intermittent RESET packets directed from the same host to apparently completely random hosts in my netblock, also directed toward TCP 98. Other than being a RESET scan, these packets have pretty much the same characteristics as the SYN packets, strongly indicating that they have been crafted by hacker toolz.

Presumably, the attacker is targeting a range of netblocks wider than just mine, and so when he scanned sequentially (the SYN scan), we saw lots of packets coming very quickly. When he randomized his scan (the RESET scan), we saw them very intermittently. Oddly, there was one RESET packet mixed in with the SYN packets. It's not shown in the above detect, but it occurred right in the middle of the SYNs:

```
00:26:40.559739 203.177.11.196.3839 > xxx.yyy.87.105.98: R 3254028326:3254028326(0) win 35013 (ttl 61, id 27661)
```

This is strange and not entirely explainable. If it is the result of a concurrent scan process, why did it take 45 minutes for my network to get probed again, after which I saw packets every few minutes?

A reset scan normally works by expecting "ICMP host unreachable" messages back for hosts that do not exist -- hosts that don't generate a response at all are hosts worth checking out later, as they probably exist.

Since the SYN scan didn't generate any information, I'm not sure what the attacker expected to gain with a reset scan. Very likely, neither was he. Diagnosis: script kiddie.

6. Correlations:

Scanning with SYN packets to find linuxconf running on port 98 is an old and well-known attack. A web search unearthed descriptions of this going back to mid-1998. More recent descriptions can be found at http://www.sans.org/y2k/020300.htm and http://www.sans.org/y2k/012000.htm.

I was unable to find any instances of reset scans targeted against TCP port 98. This may be a new variation on an old theme.

7. Evidence of active targeting:

Active targeting is a virtual certainty here, as indicated by the rapid and sequential stepping through every class C subnet of my whole netblock, not to mention the obviously crafted packets.

8. Severity:

ELEMENT	SCORE	EXPLANATION
Criticality	5	There are many critical machines on this network, including firewalls, and application and file servers.
Lethality	3	If a machine should be detected running linuxconf, the possibility of serious compromise exists. Still, this attack was merely mapping and no actual attacks against linuxconf were seen.
Countermeasures	5	We don't run Linux at this site, and if we did we certainly wouldn't run linuxconf.
Network CM	5	Port 98 is not allowed in or out through the firewall.
TOTAL SEVERITY	-2	(5 + 3) - (5 + 5)

9. Defensive recommendation:

No additional defensive measures are needed. There are no Linux hosts at this site, and the firewall eats port 98 packets (no matter how they are constructed) without sending anything back to the originating host.

10. Multiple choice test question:

Given the following trace:

```
00:23:12.272490 203.177.11.196.1418 > xxx.yyy.0.50.98: S 3154840318:3154840318(0)
        win 32120 <mss 1460,sackOK,timestamp 6428211[|tcp]> (DF) (ttl 49, id 10850)
00:23:12.320171 203.177.11.196.1419 > xxx.yyy.0.51.98: S 3146254879:3146254879(0)
        win 32120 <mss 1460,sackOK,timestamp 6428234[|tcp]> (DF) (ttl 49, id 10853)
00:23:12.327858 203.177.11.196.1420 > xxx.yyy.0.52.98: S 3151667122:3151667122(0)
        win 32120 <mss 1460,sackOK,timestamp 6428235[|tcp]> (DF) (ttl 49, id 10854)
00:23:12.336925 203.177.11.196.1421 > xxx.yyy.0.53.98: S 3147479060:3147479060(0)
        win 32120 <mss 1460,sackOK,timestamp 6428235[|tcp]> (DF) (ttl 49, id 10855)
00:23:12.345286 203.177.11.196.1422 > xxx.yyy.0.54.98: S 3159665039:3159665039(0)
        win 32120 <mss 1460,sackOK,timestamp 6428236[|tcp]> (DF) (ttl 49, id 10857)
```

What indications exist that these packets are crafted?

a) The closely related values in the timestamp field in the TCP options.

b) The presence of an mss value along with sackOK in the TCP options.

c) The anomalously low TTL value.

d) None of the above.

Answer: d)

Last Updated: January 23rd, 2019

Upcoming Training

SANS Security East 2019	New Orleans, LA	Feb 02, 2019 - Feb 09, 2019	Live Event
Security East 2019 - SEC503: Intrusion Detection In-Depth	New Orleans, LA	Feb 04, 2019 - Feb 09, 2019	vLive
SANS Northern VA Spring- Tysons 2019	Tysons, VA	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS London February 2019	London, United Kingdom	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS Scottsdale 2019	Scottsdale, AZ	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS New York Metro Winter 2019	Jersey City, NJ	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201902,	Feb 27, 2019 - Apr 04, 2019	vLive
SANS San Francisco Spring 2019	San Francisco, CA	Mar 11, 2019 - Mar 16, 2019	Live Event
SANS Madrid March 2019	Madrid, Spain	Mar 25, 2019 - Mar 30, 2019	Live Event
SANS 2019	Orlando, FL	Apr 01, 2019 - Apr 08, 2019	Live Event
Blue Team Summit & Training 2019	Louisville, KY	Apr 11, 2019 - Apr 18, 2019	Live Event
SANS Riyadh April 2019	Riyadh, Kingdom Of Saudi Arabia	Apr 13, 2019 - Apr 18, 2019	Live Event
Community SANS New York SEC503	New York, NY	Apr 29, 2019 - May 04, 2019	Community SANS
SANS Security West 2019	San Diego, CA	May 09, 2019 - May 16, 2019	Live Event
SANS Northern VA Spring- Reston 2019	Reston, VA	May 19, 2019 - May 24, 2019	Live Event
SANS Amsterdam May 2019	Amsterdam, Netherlands	May 20, 2019 - May 25, 2019	Live Event
SANS San Antonio 2019	San Antonio, TX	May 28, 2019 - Jun 02, 2019	Live Event
San Antonio 2019 - SEC503: Intrusion Detection In-Depth	San Antonio, TX	May 28, 2019 - Jun 02, 2019	vLive
SANS London June 2019	London, United Kingdom	Jun 03, 2019 - Jun 08, 2019	Live Event
SANSFIRE 2019	Washington, DC	Jun 15, 2019 - Jun 22, 2019	Live Event
Security Operations Summit & Training 2019	New Orleans, LA	Jun 24, 2019 - Jul 01, 2019	Live Event
SANS Paris July 2019	Paris, France	Jul 01, 2019 - Jul 06, 2019	Live Event
SANS Rocky Mountain 2019	Denver, CO	Jul 15, 2019 - Jul 20, 2019	Live Event
SANS Columbia 2019	Columbia, MD	Jul 15, 2019 - Jul 20, 2019	Live Event
SANS Boston Summer 2019	Boston, MA	Jul 29, 2019 - Aug 03, 2019	Live Event
SANS Chicago 2019	Chicago, IL	Aug 19, 2019 - Aug 24, 2019	Live Event
SANS Copenhagen August 2019	Copenhagen, Denmark	Aug 26, 2019 - Aug 31, 2019	Live Event
SANS Oslo September 2019	Oslo, Norway	Sep 09, 2019 - Sep 14, 2019	Live Event
SANS Network Security 2019	Las Vegas, NV	Sep 09, 2019 - Sep 16, 2019	Live Event
SANS London September 2019	London, United Kingdom	Sep 23, 2019 - Sep 28, 2019	Live Event
SANS OnDemand	Online	Anytime	Self Paced