GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at http://www.giac.org/registration/gcia

SANS/GIAC Practical Assignment
Ray Harnes

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

GIAC Practical Assignment by Ray Harnes

Detect 1

```
195.76.27.44 > MY.NET.1.1
23:00:08.268287 195.76.27.44.65535 > MY.NET.1.1.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:00:08.271916 195.76.27.44.65535 > MY.NET.1.2.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:00:08.294812 195.76.27.44.65535 > MY.NET.1.3.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:00:08.317808 195.76.27.44.65535 > MY.NET.1.4.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:00:08.330230 195.76.27.44.65535 > MY.NET.1.5.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:21:38.484746 195.76.27.44.65535 > MY.NET.254.246.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:21:38.497681 195.76.27.44.65535 > MY.NET.254.247.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:21:38.510206 195.76.27.44.65535 > MY.NET.254.248.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:21:38.531702 195.76.27.44.65535 > MY.NET.254.249.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
23:21:38.548972 195.76.27.44.65535 > MY.NET.254.250.53:
S 2047410176:2047410176(0) win 512 (ttl 241, id 31241)
```

1. **Source of Trace**: http://www.sans.org/y2k/060300.htm
2. **Detect was generated by**: tcpdump or windump
3. **Probability that the IP address was spoofed**: probably not. This was a fast network scan looking for a DNS server. The attacker would need to see the reply packets to find the server.
4. **Description of attack**: Attack against TCP port 53 (DNS). This is a network scan, searching for a DNS box. Looks like a crafted packet since the source port is always 65535. Also the attacker is scanning multiple all hosts on multiple subnets.
5. **Attack Mechanism**: This attack works by sending a syn packet to port 53 on each host. If there is no syn-ack sent back there is no host at that address, also if the connection is refused the box is not hosting DNS. If the attacker gets back the proper response they know DNS is present. The attacker could then work towards obtaining a zone transfer or committing a denial of service attack.
6. **Correlation**: This detect was attributed by David Hoelzer. This is a common type of network scan.
7. **Evidence of Active Targeting**: There is no evidence of active targeting here. The attacker is scanning multiple subnets looking for a specific service to exploit.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (5 + 3) – (3 + 3) = Severity 2
   Assumed moderate countermeasures on both the system and the net. There is only so much that can be done because we can't entirely filter DNS requests. It would be good to have a firewall in place with rules to allow only certain hosts or networks access your DNS server.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

9. **Defensive Recommendation**: It would be good to have a firewall in place with rules to allow only certain hosts or networks access your DNS server. Also filtering of high-speed scans like this would be a good idea. Either can be done with a good packet filter of stateful firewall.
10. **Multiple choice test question**:

    a) DNS Zone Transfer
    b) Network mapping
    c) DNS Buffer Overflow
    d) Network Scan

    Answer: d

Detect 2

```
June 1 08:34:25.118877 210.196.222.18.53 > MyNet.3.53:
  SF 907639663:907639663(0) win 1028
June 1 08:34:25.140392 210.196.222.18.53 > MyNet.4.53:
  SF 907639663:907639663(0) win 1028
June 1 08:34:25.164549 210.196.222.18.53 > MyNet.5.53:
  SF 907639663:907639663(0) win 1028
June 1 08:34:25.174844 210.196.222.18.53 > MyNet.6.53:
  SF 907639663:907639663(0) win 1028
June 1 08:34:25.193856 210.196.222.18.53 > MyNet.7.53:
  SF 907639663:907639663(0) win 1028
June 1 08:34:25.218688 210.196.222.18.53 > MyNet.8.53:
  SF 907639663:907639663(0) win 1028
June 1 08:34:25.242696 210.196.222.18.53 > MyNet.9.53:
  SF 907639663:907639663(0) win 1028
```

1. **Source of Trace**: http://www.sans.org/y2k/060300.htm
2. **Detect Generated By**: Windump
3. **Probability source address was spoofed**: Not likely. This looks like a syn-fin network scan. The attacker would need to receive the responses for the network scan to be worth doing.
4. **Description of Attack**: This is a typical network scan. Instead of using ICMP packets the attacker has crafted syn-fin packets to map which hosts are alive and which are not. Also this attacker is trying to hit port 53 (DNS). He could be looking for a DNS server (which is why they would hit 53) and/or just mapping the network (a SF combination would cause the remote host to send a R). Each time a machine sends a Reset, the attacker knows there is a host alive at that address. The SF flags both being set are believed to fool packet filters and are "supposedly" impossible to log.
5. **Attack Mechanism**: The attacker could be looking for a DNS server (which is why they would hit 53) and/or just mapping the network (a SF combination would cause the remote host to send a R). Each time a machine sends a Reset, the attacker knows there is a host alive at that address. The SF flags both being set are believed to fool packet filters and are "supposedly" impossible to log.

© SANS Institute 2000 - 2002, Author retains full rights.

6. **Correlation**: Judith M. Ostroot. This is a common network mapping technique.
7. **Evidence of Active Targeting**: None. Again the attacker is on a fishing expedition. Looking across the entire subnet to see what they can find.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (4 + 2) – (2 + 2) = 2
   Severity could be high if they are looking for DNS. Criticality depends on what they do after they have scanned the network. I give it a 2. Countermeasures are not obvious here other than the sniffer on the network. I would assume minimum counter measures. I would give this a severity of 2 with no further information on what was scanned after this.
9. **Defensive Recommendation**: An IDS should be employed with a filter to block all packets with SF flag from entering the network. There is no natural use for SF flags on the same packet. This points out that these are crafted.
10. **Multiple Choice Test Question**:

    a) **DNS Zone Transfer**
    b) **Network Mapping**
    c) **Denial of Service**
    d) **Syn Flood**

    Answer: b

Detect 3

```
May 29 19:32:01 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33474
May 29 19:32:04 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33475
May 29 19:32:06 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33476
May 29 19:32:08 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33477
May 29 19:32:11 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33478
May 29 19:32:15 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33479
May 29 19:32:18 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33480
May 29 19:32:21 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33481
May 29 19:32:26 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33482
May 29 19:32:28 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33483
May 29 19:33:51 Deny inbound udp src153.39.203.154/52995
  dst xxx.xxx.xxx.50/33506
```

1. **Source of the Trace**: http://www.sans.org/y2k/053100-1200.htm
2. **Detect Generated by**: Router Log file

© SANS Institute 2000 - 2002, Author retains full rights.

3. **Probability Source Address was spoofed**: Not likely. This detect is a traceroute. The attacker would need to see the ICMP time exceeded and unreachable messages in order for the traceroute to map the network
4. **Description of attack**: Attacker sends UDP packets to high order ports incrementing the TTL by 1 for each packet. This lets them know how many hops away you are and helps them map your network.
5. **Attack Mechanism**: The attack basically maps your network. Each packet sent with an incremented TTL will get one hop closer to the box the attacker is trying to hit. If the packet doesn't make it to the host a time exceeded message is sent back. Once the TTL is high enough the packet will make it to the host. If the host doesn't have a service running on the port the attacker is hitting they get back an unreachable message. This attack maps the network and the routes to the network.
6. **Correlation**: This is similar to using ICMP echo reply packets and will accomplish almost the same thing. However, the packets will get through a packet filter since a packet filter will not examine the state of the connection.
7. **Evidence of Active Targeting**: This attacker is actively targeting xxx.xxx.xxx.50.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (4 + 4) – (2 + 3) = 3
   This could be a problem. The network countermeasures are working since the packets have been denied. I think I would silently drop these packets and implement the No Unreachables command on the router.
9. **Defensive Recommendation**: Defenses seem to have worked here. The probe packets have been denied and probably dropped at the router. But this itself could give information to the attacker. No IP Unreachables.
10. **Multiple choice test question**:

    a) **Traceroute**
    b) **"Wrong Number"**
    c) **Port Scan**
    d) **Firewall-1 Log**

    Answer: c

Detect 4

Feb 27 12:42:19 morannon kernel: Packet log: input DENY eth0 PROTO=1 216.41.28.66:8 x.x.x.21:0 L=60 S=0x00 I=4034 F=0x0000 T=10 (#1) Feb 27 12:42:21 morannon kernel: Packet log: input DENY eth0 PROTO=1 216.41.28.66:8 x.x.x.21:0 L=60 S=0x00 I=4290 F=0x0000 T=10 (#1) Feb 27 12:42:22 morannon kernel: Packet log: input DENY eth0 PROTO=1 216.41.28.66:8 x.x.x.21:0 L=60 S=0x00 I=4546 F=0x0000 T=10 (#1) Feb 27 12:42:23 morannon kernel: Packet log: input DENY eth0 PROTO=1 216.41.28.66:8 x.x.x.21:0 L=60 S=0x00 I=4802 F=0x0000 T=10 (#1)

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

1. **Source of Trace**: http://www.sans.org/y2k/022900.htm
2. **Detect generated by**: Firewall Log
3. **Probability Source IP Spoofed**: High probability
4. **Description of Attack**: This looks to be an IP stack disabling denial of service attack, or an attempt to circumvent a packet filter.
5. **Attack Mechanism**: This attack looks like it will disable particular IP stacks by trying to send data to port 0. Or perhaps is an attempt to circumvent certain firewalls or packet filters by accessing Port 0. It has also been reported on BugTraq that ssh, although initially connecting to port 22, will take over port 0. However, I don't believe this to be the case here. I believe this to be a DoS attack.
6. **Correlation**: This correlates to a couple of DoS attacks I have read about as well as the idea that ssh may change to port 0 after it's initial connection. I haven't seen this in practice using ssh, but would not overrule the idea without more investigation.
7. **Evidence of Active Targeting**: All packets point to the same dst address and port 0. It looks like a DoS attack on a specific host.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (2 + 3) – (2 + 3) = 0
   This appears to be fairly benign in this example. Although port 0 may cause a service outage, the countermeasures in place took care of it by denying the packet.
9. **Defensive Recommendation**: The current defenses seem to work fine in this example.
10. **Multiple choice test question**:

    a) **Demon Internet**
    b) **Denial of Service**
    c) **Sscan Signature**
    d) **ICMP flood**

    Answer: b

Detect 5

```
May 22 11:06:27 leviathan ipmon[23080]: 11:06:26.565838 ne3
  @0:15 b 210.140.231.147,109 -> 204.245.8.48,109 PR tcp len 20 40 -SF
May 22 11:06:27 leviathan ipmon[23080]: 11:06:26.602582 ne3
  @0:15 b 210.140.231.147,109 -> 204.245.8.50,109 PR tcp len 20 40 -SF
May 22 11:06:27 leviathan ipmon[23080]: 11:06:26.862078 ne3
  @0:15 b 210.140.231.147,109 -> 204.245.8.63,109 PR tcp len 20 40 -SF
```

1. **Source of Trace**: http://www.sans.org/y2k/052600-1130.htm
2. **Detect was generated by**: Leviathan
3. **Probability source address spoofed**: No likely
4. **Description of Attack**: This looks to be a network scan for POP 2.
5. **Attack Mechanism**: This looks like network reconnaissance. A packet is sent to port 109 (POP2) with the SF flags set. The attacker has no intention of completing

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

the 3-way handshake to establish the connection, thus the F flag. This seems like he is just looking for a POP2 box to exploit.

6. **Correlation**: This was a common exploit. Both POP2 and POP3 daemons support the concept of an "anonymous proxy" - where remote users can connect and open an IMAP mailbox on any server they have a valid account on. An attacker can connect to the vulnerable POP2 daemon port and issue a command to connect to an IMAP server under their control. Once logged on, issuing a "FOLD" command with a long argument will cause a buffer overflow to buffer that resides on the stack. The argument to FOLD must be around 1000 bytes to cause the overflow.
7. **Evidence of Active Targeting**: I don't see evidence here. The attacker is still looking for a box to target.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (3 + 3) – (3 + 3) = 0
   There are not a lot of POP2 boxes left. Even though POP3 has a similar vulnerability it looks like this person is trying for a particular type of box to use an old exploit. The IDS detected this and therefore the countermeasures are working correctly. I might firewall out all POP3 connections except those from a particular network that you may want to allow.
9. **Defensive Recommendation**: Defenses are adequate. Only suggest either closing POP3 in case a similar POP3 exploit is around or firewall out disallowed networks.
10. **Multiple Choice Test Question**:

    a) **Network Scan**
    b) **POP2 Exploit**
    c) **Cisco Router Log**
    d) **Syn Flood**

Detect 6

A dialup user in UUnet space. 1cust131.tnt6.topeka.ks.da.uu.net

```
04/23 23:58:24.286949 63.21.146.131.3773 > 10.0.108.21.12345:
S 12401930:12401930(0) win 819 <mss 536,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> [tos 0xb0] (ttl 48, id 13179)

04/23 23:58:24.291482 63.21.146.131.3774 > 10.0.108.21.1243:
S 12401931:12401931(0) win 8192 <mss 536,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> [tos 0x94] (ttl 48, id 13435)

04/23 23:58:24.292103 63.21.146.131.3778 > 10.0.108.21.31337:
S 12401939:12401939(0) win 8192 <mss 536,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> [tos 0x84] (ttl 48, id 13691)

04/23 23:58:24.431566 63.21.146.131.3788 > 10.0.108.21.12346:
S 12401961:12401961(0) win 8192 <mss 536,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> [tos 0x84] (ttl 48, id 14203)

04/23 23:58:24.432815 63.21.146.131.3777 > 10.0.108.21.21554:
S 12401937:12401937(0) win 8192 <mss 536,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> [tos 0x48] (ttl 48, id 14459)
```

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
04/23 23:58:24.438045 63.21.146.131.3793 > 10.0.108.21.6969:
S 12401978:12401978(0) win 8192 <mss 536,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> [tos 0xb0] (ttl 48, id 14971)
```

1. **Source of trace**: http://www.sans.org/y2k/042900.htm
2. **Detect was generated by**: tcpdump/shadow
3. **Probability source address is spoofed**: not likely. For the scan to be effective, the attacker would need to see the replies.
4. **Description of attack**: This attacker is looking for some well-known trojans. This looks like and intentional port scan over common trojan ports.
5. **Attack Mechanism**: The attacker is sending syn packets to 10.0.28.21, on specific ports. It's not like a normal port scan. This person knows what they are looking for. Ports 31337, 6969, 12345, are really a tip off.
6. **Correlation**: This happens all the time. I can' correlate it to a specific attack since it's just a fishing expedition.
7. **Evidence of active targeting**: This machine was hit multiple times on multiple ports. There is no way this is an accident. This definitely demonstrates active targeting.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (4 + 4) – (3 + 3) = 2
   The criticality and lethality could be high if the presence of a trojan was found. Here it looks like nothing is there to respond. I can't tell much about the countermeasures except that this person can sniff the wire. It's reasonable to assume that there is a firewall of IDS in place, so I put the countermeasures on the moderate side. If there was a trojan active the severity is quite high. However, if there was a trojan we should see some client server type of traffic with the 3-way handshake completing. This doesn't appear to be the case.
9. **Defensive Recommendations**: The implied deny all all rule for the firewall will help stop this kind of scan. Also want to have an IDS scream when these ports are scanned. I would also have a filter on my IDS that says if so many packets from the same address come in within a certain period of time, set off an alarm.
10. **Multiple choice test question**:

    a) **Syn flood**
    b) **Trojan Scan**
    c) **Covert channel**
    d) **Nmap**

    Answer: b

Detect 7

```
Apr 19 23:43:03 cc1014244-a kernel: securityalert: tcp if=ef0 from
172.131.63.199:4752 to xx.x.xx.xxx on unserved port 27374
Apr 20 01:25:08 cc1014244-a kernel: securityalert: tcp if=ef0 from
209.214.119.195:1710 to xx.x.xx.xxx on unserved port 30100
```

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
Apr 20 03:57:50 cc1014244-a kernel: securityalert: udp if=ef0 from
24.13.27.27:137 to xx.x.xx.xxx on unserved port 137

Apr 20 11:02:05 cc1014244-a kernel: securityalert: udp if=ef0 from
195.184.193.12:60000 to xx.x.xx.xxx on unserved port 2140

Apr 20 13:43:10 cc1014244-a kernel: securityalert: udp if=ef0 from
195.184.193.12:60000 to xx.x.xx.xxx on unserved port 2140
Apr 20 15:54:29 cc1014244-a kernel: securityalert: udp if=ef0 from
24.26.88.197:137 to xx.x.xx.xxx on unserved port 137
Apr 20 20:34:13 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2786 to xx.x.xx.xxx on unserved port 22
Apr 20 20:35:24 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2788 to xx.x.xx.xxx on unserved port 5632
Apr 20 20:35:24 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2788 to xx.x.xx.xxx on unserved port 22
Apr 20 20:39:00 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2796 to xx.x.xx.xxx on unserved port 22
Apr 20 20:44:27 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2804 to xx.x.xx.xxx on unserved port 22
Apr 20 21:26:34 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2873 to xx.x.xx.xxx on unserved port 22
Apr 20 21:27:35 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2875 to xx.x.xx.xxx on unserved port 5632
Apr 20 21:27:35 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2875 to xx.x.xx.xxx on unserved port 22
Apr 20 21:29:42 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2883 to xx.x.xx.xxx on unserved port 22
Apr 20 21:30:51 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2888 to xx.x.xx.xxx on unserved port 5632
Apr 20 21:30:51 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2888 to xx.x.xx.xxx on unserved port 22
Apr 20 21:53:32 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.21.225:2925 to xx.x.xx.xxx on unserved port 22
Apr 20 22:26:49 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.20.229:1202 to xx.x.xx.xxx on unserved port 5632
Apr 20 22:26:49 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.20.229:1202 to xx.x.xx.xxx on unserved port 22
Apr 20 22:27:05 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.20.229:1204 to xx.x.xx.xxx on unserved port 5632
Apr 20 22:27:05 cc1014244-a kernel: securityalert: udp if=ef0 from
24.3.20.229:1204 to xx.x.xx.xxx on unserved port 22
```

1. **Source of Trace**: http://www.sans.org/y2k/042400.htm
2. **Detect generated by**: Gauntlet Firewall Log
3. **Probability Address was spoofed**: Probably Not looks like trying to connect to several services. However, there are multiple addresses involved. It could be multiple attackers, or one attacker from different locations. I don't think it's a spoofed address.
4. **Description of Attack**: I believe a number of things are going on here. Port 22 could be either ssh secure shell or PC anywhere. With the presence of packets going to port 5632 it looks like PC Anywhere, but either way it is an active attempt to connect. PCanywhere consists mainly of UDP packets sent to port 5632 but will look

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

to port 22 if it can't find anything on 5632. This is for backward compatibility. .Port 2140 could be the port for a trojan called Deep Throat. Port 137 looks like a netbios nbname packet.

5. **Attack mechanism**: This looks like it could be multiple attackers on different days. Or the same attacker from a couple different locations. They are looking for trojans and remote control software.
6. **Correlation**: Just another example of scanning a machine for open ports.
7. **Evidence of active targeting**: It looks pretty active to me. The same dst address on the same ports consistently. This person may have already done some other reconnaissance.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (4 + 3) – (3 + 4) = 1
   The severity could be high but the countermeasures. Since Gauntlet is present and is running, the countermeasures seem to be pretty good. Since this went on for a couple of days I would say the attacker is pretty determined so I gave a severity of 1.
9. **Defensive recommendation**: Current defenses are working well. The firewall can silently drop these with no problem.
10. **Multiple choice test question**:

    a) **DdoS**
    b) **Low and slow**
    c) **Back Orifice**
    d) **Trojan Scan**

    Answer: d

Detect 8

```
Apr 9 00:54:05 hostp portsentry[522]: attackalert: Connect
from host: 195.145.171.21/195.145.171.21 to TCP port: 1524
Apr 9 00:54:05 hostp portsentry[522]: attackalert: Connect
from host: 195.145.171.21/195.145.171.21 to TCP port: 1524
Apr 9 00:54:05 hostr portsentry[418]: attackalert: Connect
from host: 195.145.171.21/195.145.171.21 to TCP port: 1524
Apr 9 00:54:05 hostb portsentry[334]: attackalert: Connect
from host: 195.145.171.21/195.145.171.21 to TCP port: 1524
Apr 9 00:54:49 hostc portsentry[15996]: attackalert: Connect
from host: 195.145.171.21/195.145.171.21 to TCP port: 1524
Apr 9 00:57:59 hostd portsentry[416]: attackalert: Connect
from host: 195.145.171.21/195.145.171.21 to TCP port: 1524
Apr 9 01:19:11 dns1 portsentry[438328]: attackalert: Connect
from host: 195.145.171.21/195.145.171.21 to TCP port: 1524
```

1. **Source of trace**: http://www.sans.org/y2k/041200.htm
2. **Detect was generated by**: portsentry port scan detector
3. **Probability source address was spoofed**: probably not. Although trinoo is a distributed DOS it looks as though this person was trying to find trinoo running on this machine.

© SANS Institute 2000 - 2002, Author retains full rights.

4. **Description of attack**: Attacker probing a specific machine looking for Trinoo. He has probably checked many machines as this is DdoS attack and would require a number of hosts for it to work effectively.
5. **Attack Mechanism** "1). A stolen account is set up as a repository for pre-compiled versions of scanning tools, attack (i.e. buffer overrun exploit) tools, root kits and sniffers, trinoo daemon and master programs, lists of vulnerable hosts and previously compromised hosts, etc. This would normally be a large system with many users, one with little administrative oversight, and on a high-bandwidth connection for rapid file transfer. 2). A scan is performed of large ranges of network blocks to identify potential targets. Targets would include systems running various services known to have remotely exploitable buffer overflow security bugs, such as wu-ftpd, RPC services for "cmsd", "statd", "ttdbserverd", "amd", etc. Operating systems being targeted appear to be primarily Sun Solaris 2.x and Linux (due to the ready availability of network sniffers and "root kits" for concealing back doors, etc.), but stolen accounts on any architecture can be used for caching tools and log files. 3). A list of vulnerable systems is then used to create a script that performs the exploit, sets up a command shell running under the root account that listens on a TCP port (commonly 1524/tcp, the "ingreslock" service port), and connects to this port to confirm the success of the exploit. In some cases, an electronic mail message is sent to an account at a free web based email service to confirm which systems have been compromised. The result is a list of "owned" systems ready for setting up back doors, sniffers, or the trinoo daemons or masters. 4). From this list of compromised systems, subsets with the desired architecture are chosen for the trinoo network. Pre-compiled binaries of the trinoo daemon are created and stored on a stolen account somewhere on the Internet. 5). A script is then run which takes this list of "owned" systems and produces yet another script to automate the installation process, running each installation in the background for maximum multitasking. " [1]
6. **Correlation:** Trinoo is a fairly well known distributed denial of service attack.
7. **Evidence of active targeting**: This was not a network scan. The only machine touched is listed. Very active targeting.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (3 + 3) – (1 + 3) = 2
   The severity here is keyed to multiple machines. Finding them scanning one machine on the network may not be crucial, but if you see many more, it could be that you have been compromised and used for a DdoS.
9. **Defensive Recommendation**: Portsentry caught the scan, which means net countermeasures worked. However, we need to check our network for machines that respond on port 1524. If there are others we need to make sure that trinoo isn't on site. We may want to block port 1524.
10. **Multiple choice test question**:

    a) **Trinoo**
    b) **Cisco Firewall Log**
    c) **ftp**
    d) **network scan**

[1] 1. Description of Trinoo from http://www2.merton.ox.ac.uk/~security/bugtraq-199912/0094.html

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Detect 9

Mar 31 19:09:35 hosth snort[75541]: spp_portscan: PORTSCAN DETECTED from 203.85.30.129 Mar 31 19:09:41 hosth snort[75541]: spp_portscan: portscan status from 203.85.30.129: 14 connections across 14 hosts: TCP(14), UDP(0) Mar 31 19:09:47 hosth snort[75541]: spp_portscan: End of portscan from 203.85.30.129 -------- Mar 31 19:09:34 203.85.30.129:1542 -> A.B.C.30:98 SYN **S***** Mar 31 19:09:34
203.85.30.129:1545 -> A.B.C.33:98 SYN **S***** Mar 31 19:09:38
203.85.30.129:1710 -> A.B.C.197:98 SYN **S***** Mar 31 19:09:38
203.85.30.129:1714 -> A.B.C.201:98 SYN **S***** Mar 31 19:09:38
203.85.30.129:1717 -> A.B.C.204:98 SYN **S***** Mar 31 19:09:38
203.85.30.129:1720 -> A.B.C.207:98 SYN **S***** Mar 31 19:09:38
203.85.30.129:1727 -> A.B.C.214:98 SYN **S***** Mar 31 19:09:38
203.85.30.129:1728 -> A.B.C.215:98 SYN **S***** Mar 31 19:09:38
203.85.30.129:1731 -> A.B.C.218:98 SYN **S***** Mar 31 19:09:36
203.85.30.129:1748 -> A.B.C.235:98 SYN **S***** Mar 31 19:09:36
203.85.30.129:2021 -> A.B.D.252:98 SYN **S***** Mar 31 19:09:37
203.85.30.129:1531 -> A.B.C.19:98 SYN **S***** Mar 31 19:09:39
203.85.30.129:2006 -> A.B.D.237:98 SYN **S***** Mar 31 19:09:39
203.85.30.129:2073 -> A.B.E.48:98 SYN **S*****

1. **Source of trace:** http://www.sans.org/y2k/040200.htm
2. **Detect was generated by**: looks like snort and tcpdump
3. **Probability source address is spoofed**: probably not
4. **Description of attack**: This is a scan for linuxconf
5. **Attack mechanism**: The attacker is scanning multiple machines in a single subnet looking for a daemon on port 98. They are sending syn packets which will help them map out the network. Once they find a machine that responds on that port, they will have a good idea that it is a linux box. This will allow them to actively target that specific machine with some linuxconf exploit.
6. **Correlation**: This is very common at this point. There are many traces I have seen where the attacker is looking for linuxconf.
7. **Evidence of active targeting**: This attacker is not actively targeting a specific host, but he is scanning a single subnet. Or at least that's all we can see from the trace.
8. **Severity**: Severity = (criticality + lethality) – countermeasures (system + net)
   (3 + 2) – (3 + 1) = 1
   This has little severity to it. They have not located a linux host. I don't believe they can identify the OS based on the information given. They are shooting in the dark. Further, they have a sniffer and an IDS, which are very good countermeasures. I give it a 1.

9. **Defensive Recommendation**: The defenses look sufficient to me. I would add a firewall blocking incoming attempts to port 98 if there isn't one.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

10. Multiple choice test question:

a) **Syn Flood**
b) **Trojan Scan**
c) **Linuxconf**
d) **Network Mapping**

Answer: c

Detect 10

16:56:41.241712 172.22.1.201.1825 > 255.255.255.255.6666: udp 33
16:56:44.181546 172.22.1.103.3026 > 172.22.1.201.6667: P 134:241(107) ack 193 win 8342 (DF)
16:56:44.182715 172.22.1.201.6667 > 172.22.1.103.3026: P 193:351(158) ack 241 win 7916 (DF)
16:56:44.184492 172.22.1.103.3026 > 172.22.1.201.6667: P 241:268(27) ack 351 win 8184 (DF)
16:56:44.184942 172.22.1.201.6667 > 172.22.1.103.3026: P 351:385(34) ack 268 win 7889 (DF)
16:56:44.331626 172.22.1.103.3026 > 172.22.1.201.6667: . ack 385 win 8150 (DF)

"1" "10Jun2000" "17:07:10" "El90x1" "Guardian" "log" "drop" "2997" "Guardian" "172.22.1.103" "udp" "15" "6666" "" "" "" "Guardian" "172.22.1.103" "6666" "2997" " len 85"
"2" "10Jun2000" "17:07:39" "El90x1" "Guardian" "log" "drop" "1825" "Guardian" "172.22.1.201" "udp" "15" "6666" "" "" "" "Guardian" "172.22.1.201" "6666" "1825" " len 85"
"3" "10Jun2000" "17:08:15" "El90x1" "Guardian" "log" "drop" "2997" "Guardian" "172.22.1.103" "udp" "15" "6666" "" "" "" "Guardian" "172.22.1.103" "6666" "2997" " len 85"
"4" "10Jun2000" "17:08:43" "El90x1" "Guardian" "log" "drop" "1825" "Guardian" "172.22.1.201" "udp" "15" "6666" "" "" "" "Guardian" "172.22.1.201" "6666" "1825" " len 85"
"5" "10Jun2000" "17:09:20" "El90x1" "Guardian" "log" "drop" "2997" "Guardian" "172.22.1.103" "udp" "15" "6666" "" "" "" "Guardian" "172.22.1.103" "6666" "2997" " len 85"
"6" "10Jun2000" "17:09:47" "El90x1" "Guardian" "log" "drop" "1825" "Guardian" "172.22.1.201" "udp" "15" "6666" "" "" "" "Guardian" "172.22.1.201" "6666" "1825" " len 85"

1. **Source of trace**: My network
2. **Detect was generated by**: FW1 logs with windump window.
3. **Probability source IP spoofed**: None

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

4. **Description of attack**: It looked at first like my network had been compromised. I was really surprised to see these machines communicating on port 6666 and other weird ports. It looked as though IRC was running within my LAN.
5. **Attack Mechanism**: There really was no attack here. I found the firewall logs with many references to port 6666 on one of my mail servers and one workstation. It didn't look like a client server type of connection because it was UDP. Still the ports were strange. I watched for some time and found that these machines were both hosts running the Powerchute software. All of these packets are generated when Powerchute client connects to a server running Powerchute. Another thing that bogged me down was the fact that an SMTP packet kicked out a while later. It all boiled down to Powerchute was being installed and tested on these 2 machines. The test being run most often was that of a power failure kicking off an email to one of our network guys. Not quite an attack, but I spent a while analyzing it and thought it might be valuable.
6. **Correlation**: Not really sure it correlates to anything.
7. **Evidence of Active targeting**: If this had actually been an attack it looked like it was being actively targeted.
8. **Severity = 0**
9. **Defensive Recommendation**: Although this wasn't an actual attack, port 6666, and 6667 automatically sound off an alarm since they are IRC ports. Blocking all communication on the common IRC ports is a very good idea.
10. **Multiple choice test question**:

    a) **IRC**
    b) **Wrong Number**
    c) **Active Targeting**
    d) **Powerchute communications**

    Answer: d

© SANS Institute 2000 - 2002 Author retains full rights.