Robert Hew

All addresses have been changed to bogus addresses.

1) Here is a slow host sweep for the telnet port. The scan program used random destination address in addition to being a slow sweep to hide it from an intrusion detection system. The hacker was probably looking for a Unix box or network box (router/switch) to try brute force login attempts.

```
Jan 10 10:01:03 10.10.10.1:2532 -> 192.168.1.5:23 SYN **S*****
Jan 10 10:05:04 10.10.10.1:2573 -> 192.168.1.100:23 SYN **S*****
Jan 10 10:12:08 10.10.10.1:2640 -> 192.168.1.25:23 SYN **S*****
Jan 10 10:32:08 10.10.10.1:2681 -> 192.168.1.220:23 SYN **S*****
Jan 10 10:41:10 10.10.10.1:2801 -> 192.168.1.17:23 SYN **S*****
Jan 10 11:23:13 10.10.10.1:2950 -> 192.168.1.21:23 SYN **S*****
Jan 10 11:43:04 10.10.10.1:3051 -> 192.168.1.140:23 SYN **S*****
Jan 10 11:52:18 10.10.10.1:3162 -> 192.168.1.111:23 SYN **S*****
Jan 10 12:04:18 10.10.10.1:3263 -> 192.168.1.74:23 SYN **S*****
Jan 10 12:09:20 10.10.10.1:3566 -> 192.168.1.55:23 SYN **S*****
Jan 10 12:38:05 10.10.10.1:4080 -> 192.168.1.252:23 SYN **S*****
```

---

2) Here's a high udp scan. This looks like a crafted packet since the port number of the source is the same. The scan could have been looking for a listening high port for a trojan horse application.

```
Sep 15 12:06:19 10.10.10.1:14962 -> 192.168.1.22:24118 UDP
Sep 15 12:06:19 10.10.10.1:14962 -> 192.168.1.22:24119 UDP
Sep 15 12:06:19 10.10.10.1:14962 -> 192.168.1.22:24120 UDP
Sep 15 12:06:19 10.10.10.1:14962 -> 192.168.1.22:24121 UDP
Sep 15 12:06:19 10.10.10.1:14962 -> 192.168.1.22:24122 UDP
Sep 15 12:06:19 10.10.10.1:14962 -> 192.168.1.22:24123 UDP
Sep 15 12:06:20 10.10.10.1:14962 -> 192.168.1.22:24124 UDP
Sep 15 12:06:20 10.10.10.1:14962 -> 192.168.1.22:24125 UDP
Sep 15 12:06:20 10.10.10.1:14962 -> 192.168.1.22:24126 UDP
Sep 15 12:06:20 10.10.10.1:14962 -> 192.168.1.22:24127 UDP
```

---

3) Here an attacker spoofs the 10.10.10.1 network and tries to echo packets from the character generator port of several machines. The attacker either guesses that the x.x.x.1 address is the routers addresses or has already done some surveillance to determine this information.

```
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.1.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.2.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.3.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.4.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.5.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.6.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.7.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.8.1:19 UDP
Aug 3 04:28:17 10.10.10.1:7 -> 192.168.9.1:19 UDP
```

---

4) Here's a Land attack that was scripted to go down a class C network. You can tell that they
were scripted by the fast speed in which the packets were sent out.

```
Jan 7 10:01:03 192.168.1.1:139 -> 192.168.1.1:23 SYN **S*****
Jan 7 10:01:03 192.168.1.2:139 -> 192.168.1.2:23 SYN **S*****
Jan 7 10:01:03 192.168.1.3:139 -> 192.168.1.3:23 SYN **S*****
```

---

5) Here's a scripted surveillance scan for Cisco devices. They're using port 1999 the
identification port where they can sniff the packets coming back and look into the data portion
for Cisco.

```
Aug 13 15:21:38 10.10.10.1:2344 -> 192.168.1.1:1999 SYN **S*****
Aug 13 15:21:38 10.10.10.1:2344 -> 192.168.1.2:1999 SYN **S*****
Aug 13 15:21:38 10.10.10.1:2344 -> 192.168.1.3:1999 SYN **S*****
Aug 13 15:21:38 10.10.10.1:2344 -> 192.168.1.4:1999 SYN **S*****
Aug 13 15:21:39 10.10.10.1:2344 -> 192.168.1.5:1999 SYN **S*****
Aug 13 15:21:39 10.10.10.1:2344 -> 192.168.1.6:1999 SYN **S*****
```

---

6) Here's someone looking for the default Back Orifice port 31337 UDP.
Either they have planted the trojan horse on this machine or they are scanning this machine
for the program.

```
Dec 23 05:00:07 10.10.10.1:2732 -> 192.168.1.1:31337 UDP
```

---

7) Here's a single anamolous packet. The SYN and FIN bits were both set. This could have been
a crafted packet to try to confuse a tcp/ip stack resulting in a DOS attack. They might have
been targeting a particular machine that they have already discovered since this was the only
packet to come in at like this.

```
10.10.10.1:2145 > 192.168.1.1:3242: SF 3832587:3832587(0) win 512
```

---

8) Here's an icmp request that fragmented the packet. The problem with this one is the offset of
the second packet overlaps the second one.

```
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10947:1024@0+)
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10947:1024@128)
```

---

9) Here's a Microsoft scan for port 137,138,and 139. This has to be a scripted program because
of the speed of the scan. In addition to just looking for a microsoft device the hacker is also
looking for particular ports that are open.

```
Dec 27 03:12:52 10.10.10.1:8362 -> 192.168.1.1:137 SYN **S*****
Dec 27 03:12:53 10.10.10.1:8362 -> 192.168.1.1:138 SYN **S*****
Dec 27 03:12:53 10.10.10.1:8362 -> 192.168.1.1:139 SYN **S*****
Dec 27 03:12:53 10.10.10.1:8362 -> 192.168.1.2:137 SYN **S*****
Dec 27 03:12:53 10.10.10.1:8362 -> 192.168.1.2:138 SYN **S*****
Dec 27 03:12:53 10.10.10.1:8362 -> 192.168.1.2:139 SYN **S*****
```

```
Dec 27 03:12:53 10.10.10.1:8362 -> 192.168.1.3:137 SYN **S*****
Dec 27 03:12:54 10.10.10.1:8362 -> 192.168.1.3:138 SYN **S*****
Dec 27 03:12:54 10.10.10.1:8362 -> 192.168.1.3:139 SYN **S*****
```

---

10)Here's a fragment attack. Fragment packets kept coming in but no final fragment packet arrives. The hacker is trying to overload the machine with outstanding fragment packets. This will eat up the memory of the machine as it waits for the transaction to timeout.

```
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10947:1024@0+)
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10947:1024@1024+)
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10948:372@2309+)
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10949:64@2743+)
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10962:312@3475+)
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10982:128@841+)
10.10.10.1 > 192.168.1.1 icmp: echo request (frag 10991:450@582+)
```