Global Information Assurance Certification Paper

Intrusion Detects and Analysis
From the IDIC Practical Assignments
March 2000

Authors: The Intrusion Detection Immersion Curriculum Students from SANS2000 Orlando Florida

Compiled and edited by: Carla Wendt, GCIA
Introductions: Stephen Northcutt

A note from Carla Wendt, compiler of this document:
I have taken one detect from each person that scored over 81 (for a total of 56). I tried to select a wide variety. I was sure to get the ones the review panel pointed out in the notes. It was interesting to note my fellow students were a lot like me... they put more effort into the first 4 detects and slacked on the last ones... so I ended up with quite a few of the first few entries of each candidate's detects. There certainly were lots of SubSevens.

Listed below are the sources of the detects. The original reports are on the web page, www.sans.org/giac.htm, GCIA section. Certified Candidate # is first, then the number of the detect, followed by what kind of detect it is.

19-2	Low and Slow
25-2	Host scan
26-3	Ring Zero
20-5	SynFin
23-3	False Positive
24-4	Back Orifice
28-4	Internet Firewall Snoop
29 -	can't read it
30-6	Ping O Death
31-7	DNS Port Scan
32-3	Gateway Mapping
33-8	Network Mapping
34-2	Probe of ports 23, 80, 139
35-5	Ack Scan
36-1	Smurf Attack
37-10	Low and slow probe
38-3	911/Chode virus
39-3	Netware servers - netbios scan
43-1	Illegal flag combinations
44-7	UDP scan
45-6	Denial of Service
49-3	False positive

54-3	IMAP
55-8	Getting around a firewall
57-4	Queso scan
59-5	Invalid TCP flags
63-3	Majordomo list exploit
64-2	Netbus
66-1	Collect RTT figures
70-1	Recon SNMP scan
73-4	Winnuke
74-5	Targeting DMZ
75-?	RCP Port Probe
76-?	NNTP
81-4	ICQ
82-2	CGI Vulnerability
83-6	WebAmp
84-2	Compaq
88-10	Possible Port Scan
89-8	Port 1600 Scan
94-9	SNMP Write
98-5	ZoneAlarm catches Port 6970 Gate Crasher
99-2	DeepThroat
100-4	ICMP Echo Requests
104-2	DNS
105-1	UDP Port 5135
106-1	Trojan Scans
107-3	Trin00
110-1	PC Anywhere probe
111-3	What kind of firewall have we here?
112-10	"Test-cgi" web attack
113-1	Scan for unprotected shares
114-1	Streaming Media
118-8	LinuxConf
120-6	JetDirect IP Mix Up
121-2	Get File

The first trace is from a firewall, you've got to love firewall logs with their comforting "drops". What is significant here is that the analyst quickly realizes that the attacker, is another victim. If this organization's name server is actually compromised they are potentially in big trouble. When you see something like this, make the extra effort and drop a dime on the administrator. Look the system up with a whois and give them a call, after all an attacker may be reading their mail.

Host Scans

Submitted by Martin C. Walker (http://www.sans.org/y2k/practical/martin_walker.doc)

12-Mar-00	10:17:05	drop	Tcp	203.229.230.14	x.y.z.1	sunrpc	Domain
12-Mar-00	19:51:23	drop	Tcp	203.229.230.14	x.y.z.3	sunrpc	Domain
13-Mar-00	0:37:02	drop	Tcp	203.229.230.14	x.y.z.4	sunrpc	Domain
13-Mar-00	7:00:13	drop	Tcp	203.229.230.14	x.y.z.5	sunrpc	domain
18-Mar-00	0:30:22	drop	Tcp	198.109.185.2	x.y.z.1	sunrpc	638
18-Mar-00	0:30:23	drop	Tcp	198.109.185.2	x.y.z.2	sunrpc	639
18-Mar-00	0:30:24	drop	Tcp	198.109.185.2	x.y.z.3	sunrpc	640
18-Mar-00	0:30:25	drop	Tcp	198.109.185.2	x.y.z.4	sunrpc	641
18-Mar-00	0:30:26	drop	Tcp	198.109.185.2	x.y.z.5	sunrpc	642
18-Mar-00	16:31:35	drop	Tcp	207.79.139.5	x.y.z.1	sunrpc	sunrpc
18-Mar-00	16:31:35	drop	Tcp	207.79.139.5	x.y.z.2	sunrpc	sunrpc
18-Mar-00	16:31:35	drop	Tcp	207.79.139.5	x.y.z.3	sunrpc	sunrpc
18-Mar-00	16:31:35	drop	Tcp	207.79.139.5	x.y.z.4	sunrpc	sunrpc
18-Mar-00	16:31:35	drop	Tcp	207.79.139.5	x.y.z.5	sunrpc	sunrpc

Here we see the trace of three distinct host scans against the rpc portmapper of the IP addresses that appear in the DMZ network of the target site. Each of these scans has a different signature. These are recon attempts and the possible prelude to an attack. There is no history of attack from any of these IP ranges.

The first group is from ns.neo-com.net. This is almost certainly a name server and probably one that has been compromised. The cracker is using it to launch attacks while remaining anonymous. This is a “low and slow” scan, there is a large pause between the probe of each host. The cracker is hoping to pass under the threshold of whatever IDS the target site is running, unfortunately for him I personally inspect each dropped connection attempt. Note also the source port is port 53 (DNS). Monitoring of DNS is frequently turned off on IDS and firewalls due to the high level of traffic and high rate of false positives. State insensitive devices may leave this port open. The cracker is hoping that his traffic will go ignored and/or be allowed through whatever firewall or screening router exists based on the source port.

Classification: Targeted, malicious and high risk. The risk is high because the fact it is a low and slow scan coupled with the possibility of an already compromised machine as the source indicates a cracker who knows his trade.

Follow up: Notify name server administrator.

David Hoelzer, GCIA has been a fireball since his completion of the first IDIC. He has accounted for a number of significant posts on GIAC. This trace from his student practical shows on more indication of a major increase in scanning for NetBIOS NS. As I write this, May 04, 2000 we do not know the why, but we certainly know the what.

Submitted by David Hoelzer (http://www.sans.org/y2k/practical/David_Hoelzer.doc)

```
10:14:11.237886 208.180.41.86.137 > XXX.XXX.26.1.137: udp 50 (ttl 115, id 31326)
10:14:12.732806 208.180.41.86.137 > XXX.XXX.26.1.137: udp 50 (ttl 115, id 33374)
10:14:14.253833 208.180.41.86.137 > XXX.XXX.26.1.137: udp 50 (ttl 115, id 33630)
10:14:15.838358 208.180.41.86.137 > XXX.XXX.26.2.137: udp 50 (ttl 115, id 33886)
10:14:17.338931 208.180.41.86.137 > XXX.XXX.26.2.137: udp 50 (ttl 115, id 34142)
10:14:18.851362 208.180.41.86.137 > XXX.XXX.26.2.137: udp 50 (ttl 115, id 34398)
10:14:20.566938 208.180.41.86.137 > XXX.XXX.26.3.137: udp 50 (ttl 115, id 34654)
10:14:22.061870 208.180.41.86.137 > XXX.XXX.26.3.137: udp 50 (ttl 115, id 34910)
10:14:23.569899 208.180.41.86.137 > XXX.XXX.26.3.137: udp 50 (ttl 115, id 36958)
10:14:25.123754 208.180.41.86.137 > XXX.XXX.26.4.137: udp 50 (ttl 115, id 37214)
.
.
.
10:33:21.202715 208.180.41.86.137 > XXX.XXX.26.251.137: udp 50 (ttl 115, id 55652)
10:33:22.709466 208.180.41.86.137 > XXX.XXX.26.251.137: udp 50 (ttl 115, id 55908)
10:33:24.277146 208.180.41.86.137 > XXX.XXX.26.252.137: udp 50 (ttl 115, id 56164)
10:33:25.771723 208.180.41.86.137 > XXX.XXX.26.252.137: udp 50 (ttl 115, id 56420)
10:33:27.279594 208.180.41.86.137 > XXX.XXX.26.252.137: udp 50 (ttl 115, id 56676)
10:33:28.832276 208.180.41.86.137 > XXX.XXX.26.253.137: udp 50 (ttl 115, id 56932)
10:33:30.330168 208.180.41.86.137 > XXX.XXX.26.253.137: udp 50 (ttl 115, id 57188)
10:33:31.834631 208.180.41.86.137 > XXX.XXX.26.253.137: udp 50 (ttl 115, id 59236)
10:33:33.392656 208.180.41.86.137 > XXX.XXX.26.254.137: udp 50 (ttl 115, id 59492)
10:33:34.895166 208.180.41.86.137 > XXX.XXX.26.254.137: udp 50 (ttl 115, id 59748)
10:33:36.413333 208.180.41.86.137 > XXX.XXX.26.254.137: udp 50 (ttl 115, id 60004)
```

Description:

In a matter of 15 minutes someone marched through an entire Class C looking for visible shares/machines.

History/Background/Methods:

This definitely appears to be a host scan, possibly for unprotected shares or potentially in an attempt to identify Windows machines for some other purpose. This extremely noisy scan has the pattern of sending three packets to each member of the Class C address space. The scan appears to be automated since the timing is extremely consistent (2 seconds or so between packets sequentially). The UDP TTL value is somewhat interesting as is the ID number. The ID increments regularly by 256, which seems to indicate that this machine isn't doing anything else network wise except for scanning us. This tends to indicate an extremely focussed scan, not just a random poke. The TTL value is rather high (115). This is of some interest since the default UDP TTL for Windows IP implementations is 32 (see http://www.map.ethz.ch/ftp-probleme.htm). This default can be adjusted via the registry, but it does seem odd. My guess would be that they are either using a prepackaged tool that makes the adjustment, they have done a fair amount of scanning and set the value manually (this seems unlikely since the scan is so noisy), or they are running the scan from some other type of OS. A traceroute to the originating site shows 15 hops, which would reasonably put the original TTL value between 128 and 130.

This particular scan is slightly different from the Port 137 Netbios Name Query scan listed in the ArachNIDS database (IDS177) in that the source port is consistently 137. The data payload of 50 bytes is about the right size for a legitimate name query, so it seems unlikely that there is any other hidden activity occurring.

While this does appear that we are the target of an extremely focussed scan in this instance, it is of some comfort to note that the Class C "26" does not exist internally, so they are shooting blind.

A Whois trace revealed that this address space is owned by a Texas ISP and appears to be a part of their service pool. (www.tcac.net, www.tca.net, www.cox-internet.com).

This scan took place over 15 minutes and then vanished. We have detected no further activity from this site.

Threat: Low Severity: Low
Intent: Identify internal Windows hosts/shares?
Subsequent Action: None required. These packets were detected externally and were dropped silently at the firewall.

Bryce Alexander, GCIA wrote the following analysis to help analysts understand NBTSTAT.

Summary:

There has been an increase in scanning for port 137 (Netbios SMB service). This has two sources, an increase in awareness among script kiddies of the ability to discover information about a target host using NBTSTAT and the spread of an internet worm known as network.vbs.

Background:

Beginning around the first of April, 2000 there has been an increase in the number of port 137 scans reported by contributors to GIAC. At first there was concern that this was the signature of the "911" bat-chode virus/worm, fortunately that worm was written to scan a small number of network subnets and was contained rapidly.

The increases in port 137 scans were noticed in large numbers on IP segments not included in the subnets targeted by the 911 worm. All use standard Netbios "nbstat" frames, which will elicit a node status response from Netbios and SAMBA clients. This response contains a listing of any Netbios names known to that node. A typical trace of this request looks like the following packets as captured by SNORT.

```
[**] SMB Name Wildcard [**]
05/10-18:08:05.359797 badguy.com:137 -> goodguy.com:137
UDP TTL:119 TOS:0x0 ID:45361
Len: 58
00 D4 00 00 00 01 00 00 00 00 00 00 20 43 4B 41  ............ CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAAA..!
00 01                                            ..

[**] SMB Name Wildcard [**]
05/10-18:08:06.848941 badguy.com:137 -> goodguy.com:137
UDP TTL:119 TOS:0x0 ID:45617
Len: 58
00 D8 00 00 00 01 00 00 00 00 00 00 20 43 4B 41  ............ CKA
```

```
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAAA..!
00 01                                            ..
```

```
[**] SMB Name Wildcard [**]
05/10-18:08:08.348991 badguy.com:137 -> goodguy.com:137
UDP TTL:119 TOS:0x0 ID:45873
Len: 58
00 DA 00 00 00 01 00 00 00 00 00 00 20 43 4B 41  ............ CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAAA..!
00 01                                            ..
```

A decode of the Netbios data in the first packet reveals the following:

Bytes 0 & 1: Xid
Value: 00 D4 (this value increments with each new query)
Bytes 2 & 3: Opcode NMflags & Rcode
Value: 00 10 = request, query, broadcast/multicast
Bytes 4 & 5: QDcount (number of name queries in packet)
Value: 00 01 = 1 name query
Bytes 6 to 11: ANcount, NScount, ARcount
Value: 00 00 00 00 00 00 = Not used in this frame.
Byte 12: Size of name field
Value: 0x20 = decimal value 32 (next 32 bytes used for name)
bytes 13 to 45: Name field
Value 43 4b 41 41 41...(ETC.) This is the ascii string CKAAAAA...
in the packet. It is a mangled name done by splitting the hex value
of each character into two parts(nibbles) and then adding 0x41 to each nibble.
In this packet the name is an asterisk "*" followed by nulls. The
hex value of * is 2A, splitting and adding it would become:
(2+41=43) and (A+41=4B) The Ascii Value of these two results is "CK".
The remaining nulls added to 41 remain 41 or "A"
Bytes 46 Null field delimeter
Bytes 47 & 48 Question_type
Value: 00 21 = Node Status request (nbstat).
Bytes 49 & 50 Question Class
Value: 00 01 = Internet Class.

This particular trace was crafted by using the windows command:
NBTSTAT -A (Target IP Address)

Conclusions:

So why the sudden upsurge in scans across the entire network? Certainly the script kiddies have known of this command for some time and have used it for network enumeration.

The answer appears to be twofold. First there has been recent discussion on the newsgroup alt.hackers.malicious about Netbios over IP. One regular in particular (LocoHost) has posted the applicable RFC (RFC1002) and commentary on possible exploits of Netbios. Some of these posts discussed using the NBTSTAT command for learning any names associated with a target. This would tend to popularize the use of the exploit as people who follow this group will try it on their own as well as discuss their discoveries on IRC channels. The second reason appears to be due to a recent Internet worm known as network.vbs and it's derivatives (see: http://www.cert.org/incident_notes/IN-2000-02.html)

A network trace of the network.vbs worm can be found at: http://www.sans.org/y2k/honeypot_catch.htm

The infection process begins with a nbstat request frame that is indistinguishable from the frames described in this document (above). When the nbstat is answered the worm will follow it with a TCP session on port 139 which will attempt to mount to a share which is named "C" and has no password. If successful the worm will then load itself and other payload files onto various subdirectories of the victim including the startup directory. In most cases this worm is minimaly damaging as it's primary purpose has been to replicate itself. Some of the variants have been known to have other payload files, which may not be as innocent.

A search of newsgroups on Dejanews using the string "network.vbs" elicited over 11,000 matches. A random sampling of these messages indicate that people are discovering this worm on their PC's and are primarily experiencing slow response time as the PC is busy scanning for other systems to replicate itself to. This is an indication that the worm has been sucessful in replicating itself to a large number of systems.

An interesting side effect of this worm has been a rather strange pattern that periodically shows up in the scans for port 137. This pattern shows simultanious scanning from two addresses, one a legitimate address and one a private (RFC1918) address. It is my speculation that this is caused by systems that are providing proxy services on cable modems in order to share a single IP address on a cable modem. The internal (private) address is leaking out onto the network, most likely due to sharing a single ethernet hub for both internal and external interfaces.

Apr 21 00:17:29 myhost snort: SMB Name Wildcard: 192.168.0.1:137 -> my.ip.addr:137
Apr 21 00:17:29 myhost snort: SMB Name Wildcard: 24.28.135.131:137 -> my.ip.addr:137
Apr 21 00:17:31 myhost snort: SMB Name Wildcard: 24.28.135.131:137 -> my.ip.addr:137
Apr 21 00:17:31 myhost snort: SMB Name Wildcard: 192.168.0.1:137 -> my.ip.addr:137

Examination of the packets show identical ttl values and ID fields are close in value indicating a very high likelihood of originating from the same host. This is further evidence of a script driving the scanning.

Bryce Alexander

Sources:

Detects are from my own network.
Newsgroup discussions from Dejanews search engine www.deja.com .
http://www.cert.org/incident_notes/IN-2000-02.html
http://www.sans.org/y2k/honeypot_catch.htm
RFC1002

To understand the full importance of this trace, you have to look at the dates. Dirk has managed to correlate a trace from a single host across nearly two weeks. That sounds easy until you try it!

Ring Zero

Submitted by Dirk Lehmann

```
Nov 23 01:45:57 sneak kernel: Packet log: bad-dmz DENY eth1 PROTO=6 172.154.4.145:1251
172.102.45.46:8080 L=44 S=0x00 I=65026 F=0x4000 T=6 SYN (#8)
Nov 23 01:46:00 sneak kernel: Packet log: bad-dmz DENY eth1 PROTO=6 172.154.4.145:1251
172.102.45.46:8080 L=44 S=0x00 I=10755 F=0x4000 T=6 SYN (#8)
Dec  7 03:00:27 sneak kernel: Packet log: bad-dmz DENY eth1 PROTO=6 172.154.4.145:4433
172.102.45.230:8080 L=44 S=0x00 I=23345 F=0x4000 T=7 SYN (#8)
Dec  7 03:00:53 sneak kernel: Packet log: bad-dmz DENY eth1 PROTO=6 172.154.4.145:4487
172.102.45.230:3128 L=44 S=0x00 I=39729 F=0x4000 T=7 SYN (#8)
Dec  7 03:00:56 sneak kernel: Packet log: bad-dmz DENY eth1 PROTO=6 172.154.4.145:4487
172.102.45.230:3128 L=44 S=0x00 I=51761 F=0x4000 T=7 SYN (#8)
Dec  7 03:01:03 sneak kernel: Packet log: bad-dmz DENY eth1 PROTO=6 172.154.4.145:4487
172.102.45.230:3128 L=44 S=0x00 I=63537 F=0x4000 T=7 SYN (#8)
Dec  7 03:01:16 sneak kernel: Packet log: bad-dmz DENY eth1 PROTO=6 172.154.4.145:4487
172.102.45.230:3128 L=44 S=0x00 I=10290 F=0x4000 T=7 SYN (#8)


Analysis:
 Technique:
    + Automated
    + slow, each target and port pair scanned at a different hour
    + several seconds between connections to same target
    + TCP scan
    + always same source port 4487
 Intent:
    + looking for proxies / Ring Zero
 Active Targeting:
    + Yes, probably large network scan but only few systems show up in this log

 Result:
    Looks like RingZero is running on 172.154.4.145.
```

Many organizations are using the Internet to collect information on sites. I have been burned by pkware myself, in my case the internal systems that installed it were FTPing out of my site when they were rebooted to gather banner ads. In this report, Robert Hunt, GCIA, shows active intelligence gathering on systems that have installed pkware.

Scan by marketing firm

Submitted by Robert Hunt (http://www.sans.org/y2k/practical/Robert_Hunt.doc)

```
192.168.108.35 -> 149.1.1.1    IP  D=149.1.1.1 S=192.168.108.35 LEN=28, ID=38888
  149.1.1.1 -> 192.168.108.35 IP  D=192.168.108.35 S=149.1.1.1 LEN=28, ID=55190
192.168.108.35 -> 149.1.1.1    IP  D=149.1.1.1 S=192.168.108.35 LEN=28, ID=39144
  149.1.1.1 -> 192.168.108.35 IP  D=192.168.108.35 S=149.1.1.1 LEN=28, ID=49827
```

192.168.51.182 -> 149.1.1.1 IP D=149.1.1.1 S=192.168.51.182 LEN=28, ID=5138
149.1.1.1 -> 192.168.51.182 IP D=192.168.51.182 S=149.1.1.1 LEN=28, ID=26362
192.168.51.182 -> 149.1.1.1 IP D=149.1.1.1 S=192.168.51.182 LEN=28, ID=5394
149.1.1.1 -> 192.168.51.182 IP D=192.168.51.182 S=149.1.1.1 LEN=28, ID=30726

Type of detect:
Internet Firewall Snoop

Active targeting: yes

Intent:
Information gathering by Internet marketing company

Technique:
192.168 addresses are internal
149.1.1.1 is owned by PSInet (ISP)
Owners of the internal addresses are not intentionally sending ICMP requests to 149.1.1.1
From doing machine scans of each internal machine that matches this detect I find that each contains PKZIP for windows. Visiting the PKWARE website there is an FAQ which describes a function in their shareware version of PKZIP for windows. This function will apparently attempt to connect back to a sponsoring website to provide information using tsadbot.exe, an info gathering tool. I believe this ICMP traffic is said connection.

History of detect:
Various internal addresses, as far back as our detects go.

Conclusion:
Does not appear to be malicious. Is a software control issue, internally. Have asked that users not install shareware version of this software. Have written network logon script that will detect and remove the executable from the users system. Will continue to monitor for related traffic.

INFOCON status: yellow

SYN/FIN is a classic signature and analysts should be aware of this in its many forms. SYN/FIN together is illegal of course, you can't start and stop a connection at the same time. This combination is used in stack analysis or "TCP Fingerprinting", but has also been used in probing and exploit tools for several years. This pattern has historically been from source port zero, but in March 2000 we saw a reflexive version where the source was the same as the destination with POPII source port 109 to destination port 109. Another common example of this can be seen in the May 2, 2000, GIAC post with telnet source port 21 to destination port 21. Analyst Matt Fearnow describes a rarely seen variant, SF to destination port 3128 from source 3128.

SYN FIN

Submitted by Matthew Fearnow - http://www.sans.org/y2k/practical/Matt_Fearnow.doc

16:12:48.640443 proxy.scanner.com.3128 > 192.168.1.140.3128: SF 1638842970:1638842970 (0) win 1028
16:12:48.709812 proxy.scanner.com.3128 > 192.168.1.141.3128: SF 1638842970:1638842970 (0) win 1028
16:12:49.191528 proxy.scanner.com.3128 > 192.168.1.166.3128: SF 1337861729:1337861729 (0) win 1028
16:12:49.399577 proxy.scanner.com.3128 > 192.168.1.177.3128: SF 1337861729:1337861729 (0) win 1028
16:12:49.439757 proxy.scanner.com.3128 > 192.168.1.179.3128: SF 1337861729:1337861729 (0) win 1028

Active Targeting: Yes

History: No previous history.

Technique: This appears to be a scripted network scan with minimal time difference and static source port; sequence numbers change once during the scan but do not change with each packet. Note, too, that the both the SYN and FIN flags are set.

Intent: The source computer is scanning the network probably looking for hosts and hoping to get a reset back from the host. Also, note that the destination port is TCP 3128, which is what the squid proxy server runs on.

One of the things we try to teach in the intrusion detection courses is never to rely only on the destination port for analysis. Before we call it a duck, it should quack and have feathers. This trace is one example of a false positive based on port numbers. This is a very common problem.

FALSE POSITIVE
Submitted by Christopher Paul - http://www.sans.org/y2k/practical/christopher_paul.txt

```
Background:
The following trace was detected by 'Snort' running on the firewall
connected to a cable modem. The network contains only a single Internet
IP address and utilizes DNS servers outside of the internal network.

Severity: None

Summary: False positive

Trace:
[**] TeleCommando [**]
03/27-12:45:27.485002 8:0:3E:7:DB:F7 -> xx:xx:xx:xx:xx:xx type:0x800 len:0x55
dns.outside.com:53 -> my.firewall.com:61446 UDP TTL:254 TOS:0x0 ID:27303
DF Len: 51

Analysis:
False Positive. The source IP is an DNS server outside of the firewall.
The UDP:53 request coming from the inside just happened to be on a
potential trojan port (TeleCommando).

Would recommend an additional Snort rule to pass incoming udp port 53 from
both the primary and secondary dns servers.
```

Back Orifice is still one of the most common scans detected on the Internet. Many administrators and analysts become inured to this, they have seen it so often, they consider it harmless. Keep in mind that Back Orifice was used as the remote system administration tool to install the Winoo Distributed Denial of Service tools found at three Universities in the spring of 2000. That is, one of the oldest attacks against Windows systems was used to install the newest. This is evidence we can never afford to let our guard down.

BACK ORIFICE
Submitted by Eric Brock - http://www.sans.org/y2k/practical/Eric_Brock.doc
Source of logs = Firewall-1

ID	Date	Time	SourceIP	SourcePort	DestIP	DestPort	Proto	Info
74523	1Sep1999	15:46:05	dialup98.gent.skynet.be	1107	10.10.1.1	31337	udp	len 46
74524	1Sep1999	15:46:05	dialup98.gent.skynet.be	1107	10.10.1.2	31337	udp	len 46
74525	1Sep1999	15:46:05	dialup98.gent.skynet.be	1107	10.10.1.3	31337	udp	len 46
…	…	…	…	…	…	…	…	…
74775	1Sep1999	15:46:19	dialup98.gent.skynet.be	1107	10.10.1.253	31337	udp	len 46
74776	1Sep1999	15:46:19	dialup98.gent.skynet.be	1107	10.10.1.254	31337	udp	len 46
74777	1Sep1999	15:46:19	dialup98.gent.skynet.be	1107	10.10.1.255	31337	udp	len 46

Existence: Someone claiming to be dialup98.gent.skynet.be (Belgium) is visiting us.

History: There is no other history of this host or network visiting our network.

Techniques: The source port is constant (1107). One packet is being sent to each address in our subnet in a sequential manner. The packet are coming in fast.

Intent: The intent is to find a server on our subnet that will respond on UDP port 31337.

Targeting: Our class C subnet is being targeted, but no specific machines are being targeted.

Analysis: The constant source port of 1107 gives this scan a definite fingerprint. The visitor seems to be using a script that crafts packets and sequentially scans an entire class C subnet. This is a scan looking for a host listening on UDP port 31337. Because the Back Orifice trojan is known to use this port, it is safe to conclude that this is a trojan scan for Back Orifice.

Severity:

Component	Score	Comments
Criticality	3	No specific machines are being targeted
Lethality	5	Back Orifice is a very lethal vulnerability
System Countermeasures	4	All operating systems are running the latest patches.
Network Countermeasures	4	Firewall blocks all packets to port 31337
Severity Score	0	Severity = (Criticality + Lethality) – (system countermeasures + net countermeasures

This was a famous attack tool in its day, effective against Unix, VMS, Windows and networked printers. Most operating systems now are patched so that an unexpectedly large fragment does not work as a denial of service attack anymore. This version of the attack also uses gaps or negative offsets to try to implement the teardrop style attack. Notice the first fragment has a size of 380 bytes, but the offset in the second is 376. This means the IP stack would have to handle a negative number, this attack was also effective against older operating systems. They would render the negative number as an extremely large positive number and the fragment would be written on memory meant for some other purpose, eventually killing the system.

PING O DEATH
Submitted by David Wagner - http://www.sans.org/y2k/practical/David_Wagner.doc

```
19:01:26.540838 eth0 P 10.1.1.13 > 10.1.1.10: icmp: echo request (frag 4321:380@0+)
19:01:26.545463 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@376+)
19:01:26.549912 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@760+)
19:01:26.554150 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@1136+)
19:01:26.558606 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@1520+)
19:01:26.563061 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@1896+)
19:01:26.567553 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@2280+)
19:01:26.571587 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@2656+)
. . .
19:01:26.711181 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@62696+)
19:01:26.711533 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@63080+)
19:01:26.711886 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@63456+)
19:01:26.712239 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@63840+)
19:01:26.712591 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@64216+)
19:01:26.712943 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@64600+)
19:01:26.713298 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@64976+)
19:01:26.713648 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@65360)
```

This capture shows what is commonly known as the Ping-Of-Death. Note the maximum size of the assembled fragments is greater than the max IP packet size of 65535. Also, note that since these packets were captured outside the 10.1.1.0 network the use of 10.1.1.13 for the source address indicates that the source was spoofed. In this scenario, the attacker hopes to create a buffer-overflow on the target by sending this oversized packet. If the buffer-overflow is successful it could create a denial-of-service for clients attempting to connect to the target. Since the creation of oversized ping packets has been disabled in many operating systems, Ping-Of-Death packets may be hand-crafted or generated by many canned hacking tools.

This next detect illustrates a day in the life of a DNS server. These systems cannot be hidden and are constantly under attack. Further, these attacks succeed all too often, when you see a GIAC post with an attack from a system named ns.somewhere.net you know what has happened.

DNS SYSTEM SCAN
Submitted by John Millican - http://www.sans.org/y2k/practical/John_Millican.doc

On 3/4/2000 Laurie from .edu posted the following on GIAC:

> Mar 3 00:04:56 dns3 portsentry[301]: attackalert:
> Connect from host: max3-240.max3.hou.infohwy.com/207.90.225.240
> to TCP port: 79
> Mar 3 00:05:00 dns3 portsentry[301]: attackalert:
> Connect from host: max3-240.max3.hou.infohwy.com/207.90.225.240
> to TCP port: 143
> Mar 3 00:07:09 dns3 portsentry[301]: attackalert:
> Connect from host: max3-240.max3.hou.infohwy.com/207.90.225.240
> to TCP port: 79
> Mar 3 00:07:11 dns3 in.telnetd[11055]:
> refused connect from max3-240.max3.hou.infohwy.com
> Mar 3 00:07:17 dns3 portsentry[301]: attackalert:
> Connect from host: max3-240.max3.hou.infohwy.com/207.90.225.240
> to TCP port: 143
> Mar 3 00:07:25 dns3 in.telnetd[11131]:
> refused connect from max3-240.max3.hou.infohwy.com
> Mar 3 00:08:55 dns3 portsentry[301]: attackalert:
> Connect from host: max3-240.max3.hou.infohwy.com/207.90.225.240
> to TCP port: 79
> Mar 3 00:08:56 dns3 in.telnetd[11133]:
> refused connect from max3-240.max3.hou.infohwy.com
> Mar 3 00:09:02 dns3 portsentry[301]: attackalert:
> Connect from host: max3-240.max3.hou.infohwy.com/207.90.225.240
> to TCP port: 143
> Mar 3 00:09:09 dns3 in.telnetd[11134]:
> refused connect from max3-240.max3.hou.infohwy.com

Analysis: This is a common port scan on a DNS server for 3 vulnerable ports: telnet, finger and IMAP. It is difficult to state definitively that this is an automated scan because of the random time intervals (ranging from 1 second up to 2 minutes and 5 seconds) and random sequencing of ports. This is may be an attempt to disguise an automated scan.

Telnet is susceptible to buffer overflows when it is passed either long login names or passwords.

Versions of IMAP based on University of Washington's implementation are also vulnerable to buffer overflow exploits. Among the affected vendors are Microsoft, IBM AIX, Sun, BSD, Red Hat Linux, Caldera Linux, and Debian Linux. CERT has posted two advisories, CA-98.09.imapd and CA-97.09.imap_pop, describing these vulnerabilities. Successful use of this exploit could lead to root access.

Aside from buffer overflow exploits, there is also the possibility that this is a Trojan scan. Port 23 can host the Tiny Telnet Server (TTS) Trojan. Port 79 can be home to the FireHotker Trojan. FireHotkey affects Win95/98 systems and can spawn processes, listen, open the CD drive, etc. Port 143 is not affected by Trojans.

Classification: Targeted, unknown intent, low risk. I categorize this as low risk because Laurie seems to have her defenses up and her network was not comprised by this attack. This appears to be a script kiddie scan since it targets older vulnerabilities.

Follow up: Put max3-240.max3.hou.infohwy.com on the watch list.

There are a large variety of traceroute techniques including hiding the traceroutes in well known signatures such as RingZero. Many of these are used for one form or another of network mapping and this analyst has found one of the most famous mapping projects on the Internet.

GATEWAY MAPPING
Submitted by Tony Gillespie - http://www.sans.org/y2k/practical/Tony_Gillespie.doc

All of the following detects were taken from our Cisco syslogs.

Existence — Same IP address sends packets a few times each month. 204.178.16.36, which is registered to Bell Laboratories/Lucent Technologies, 700 Mountain Ave, Murray Hill, NJ 07974

History — This site has been sending a series of 3 udp packets multiple times a month since Jan

Techniques — The interesting part about this detect is that all of the detects occur between 0700 and 0715. The destination port is always above 33434. The detects always come in a series of 3 packets. Always has the same destination host.

Intent — Gateway mapping

Targeting — Only a single host machine is targeted, but this destination host doesn't exist.

Severity **-9** = (Criticality **0** + Lethality **1**) **-** (Sys CtrMeasures **5** + Network Ctrmeasures **5**)

Analysis — These detects were always consistent at approximately the same time of day. I was interested as to why Bell Laboratories were interested in us. Just for research I attempted to go to the source site's port 80. Sure enough there was a web page. This is part of the Internet Mapping Project. They use UDP traceroute-style path probes to build a tree showing paths that internet traffic takes. Since these are UDP traceroutes it is likely that the system is not a windows based system. They produce the fractal pictures seen in different magazines depicting internet bandwidth in varying colors. You can be asked to be removed from such traces. I would think that over time this site could have a great database that shows bottleneck vulnerabilities to sites.

```
udp  204.178.16.36(33128)   -> ?.?.?.1(33783),    Jan 10 07:06:51:  1   packet
udp  204.178.16.36(33128)   -> ?.?.?.1(34123),    Jan 10 07:06:57:  1   packet
udp  204.178.16.36(33128)   -> ?.?.?.1(35264),    Jan 10 07:07:12:  1   packet
udp  204.178.16.36(39702)   -> ?.?.?.1(33819),    Feb 5  07:10:18:  1   packet
udp  204.178.16.36(39702)   -> ?.?.?.1(35643),    Feb 5  07:09:57:  1   packet
udp  204.178.16.36(39702)   -> ?.?.?.1(37318),    Feb 5  07:10:02:  1   packet
udp  204.178.16.36(41497)   -> ?.?.?.1(34664),    Jan 13 07:04:59:  1   packet
udp  204.178.16.36(41497)   -> ?.?.?.1(35038),    Jan 13 07:05:05:  1   packet
udp  204.178.16.36(41497)   -> ?.?.?.1(35894),    Jan 13 07:05:21:  1   packet
udp  204.178.16.36(44782)   -> ?.?.?.1(33933),    Feb 12 07:11:20:  1   packet
udp  204.178.16.36(44782)   -> ?.?.?.1(35964),    Feb 12 07:10:59:  1   packet
udp  204.178.16.36(44782)   -> ?.?.?.1(36279),    Feb 12 07:11:04:  1   packet
udp  204.178.16.36(47968)   -> ?.?.?.1(33568),    Mar 30 07:07:29:  1   packet
udp  204.178.16.36(47968)   -> ?.?.?.1(33681),    Mar 30 07:07:35:  1   packet
udp  204.178.16.36(47968)   -> ?.?.?.1(33989),    Mar 30 07:07:50:  1   packet
udp  204.178.16.36(50459)   -> ?.?.?.1(33986),    Mar 4  07:14:26:  1   packet
udp  204.178.16.36(50459)   -> ?.?.?.1(34705),    Mar 4  07:14:42:  1   packet
udp  204.178.16.36(50459)   -> ?.?.?.1(36519),    Mar 4  07:14:20:  1   packet
udp  204.178.16.36(52324)   -> ?.?.?.1(33468),    Feb 22 07:13:34:  1   packet
```

```
udp  204.178.16.36(52324)   -> ?.?.?.1(33883),    Feb 22 07:13:50:  1   packet
udp  204.178.16.36(52324)   -> ?.?.?.1(35595),    Feb 22 07:13:28:  1   packet
udp  204.178.16.36(53092)   -> ?.?.?.1(35210),    Jan 24 07:08:19:  1   packet
udp  204.178.16.36(53092)   -> ?.?.?.1(35565),    Jan 24 07:08:24:  1   packet
udp  204.178.16.36(53092)   -> ?.?.?.1(37289),    Jan 24 07:08:40:  1   packet
udp  204.178.16.36(57045)   -> ?.?.?.1(33441),    Mar 1  07:14:56:  1   packet
udp  204.178.16.36(57045)   -> ?.?.?.1(34671),    Mar 1  07:14:34:  1   packet
udp  204.178.16.36(57045)   -> ?.?.?.1(34983),    Mar 1  07:14:40:  1   packet
udp  204.178.16.36(58286)   -> ?.?.?.1(34483),    Mar 21 07:04:38:  1   packet
udp  204.178.16.36(58286)   -> ?.?.?.1(36509),    Mar 21 07:04:16:  1   packet
udp  204.178.16.36(58286)   -> ?.?.?.1(36723),    Mar 21 07:04:22:  1   packet
```

Nine out of ten intrusion analysts agree on absolutely nothing! The next trace shows the problem with ambiguity. I know that I don't know that the actual purpose of this trace is and that happens fairly often. When we teach intrusion detection, we make sure there are a couple traces that don't have a defined answer, we want to prepare the students for this since it actually happens.

NETWORK PROBE
Submitted by Rob Bison - http://www.sans.org/y2k/practical/Rob_Bison.doc

History: Detects from GIAC Web Site
Evidence of Targeting: Yes
Technique: Port Scan
Intent: Network mapping
Severity Level: Low – Recon

Trace Extract

Port Scan - TCP destination port 8081
Feb 14 10:54:37 morton kernel: Packet log: input DENY eth0 **PROTO=6** 210.55.10.59:4047 63.224.27.201:**8081** L=64 S=0xD4 I=46690 F=0x4000 T=50 SYN(#66)

This trace I pulled from the GIAC Web Site, and essentially disagree with the analysis. The posted analysis indicates that the Netscape proxy server administration port can be form on TCP port 8081.

The Netscape proxy server uses two port numbers: one for the proxy server itself and another for the administration server. There are no standard port numbers for proxy servers; however, the default proxy server port is TCP/8080. The administration server is typically run on a random port number above 1024. This makes it harder for unauthorized users to determine where your administration server is. The port number for the administration server must be specified during installation. However, it is possible that the admin port could be assigned to TCP/8081.

For more information about Netscape proxy server
http://developer.netscape.com/docs/manuals/proxy

The next trace shows a reconnaissance probe of the style we call mixed metaphors! Telnet is more common on Unix systems and NetBIOS on Windows. This indicates the attacker does not yet have the detailed knowledge needed to launch a really dangerous attack. The flags observation indicates this may be part of a stack analysis process or TCP fingerprinting. Note we have our old friend SYN/FIN on the second trace.

PROBE OF PORTS 23, 80, 139
Submitted by Bill Connett - http://www.sans.org/y2k/practical/Bill_Connett.doc

No. Time Source Destination Protocol Info

54 05:45:55.7050 xxx.xxx.72.120 xxx.xxx.146.169 TCP 23 > 23 [ACK] Seq=1675637262 Ack=1761221395 Win=1028 Len=0
55 05:49:17.2159 xxx.xxx.72.120 xxx.xxx.146.169 TCP 4 > 80 [FIN, SYN] Seq=963161181 Ack=95514272 Win=1028 Len=0
56 05:49:17.3099 xxx.xxx.72.120 xxx.xxx.146.169 TCP 5 > 80 [PSH] Seq=963161181 Ack=95514272 Win=1028 Len=0
57 05:49:17.3489 xxx.xxx.72.120 xxx.xxx.146.169 TCP 5578 > 139 [SYN] Seq=1534610384 Ack=0 Win=512 Len=0
58 05:49:20.0249 xxx.xxx.72.120 xxx.xxx.146.169 TCP 5578 > 139 [SYN] Seq=1534610384 Ack=0 Win=32120 Len=0

This is an interesting set of port probes to one dest address ports 23 (telnet), 80 (http), and 139 (Netbios). The interesting thing is the variety of source ports and the variety of flags used. The sequence numbers in pairs indicate crafting as do the ack numbers. The attempt to probe port 80 and 139 twice each indicates a less than stealth approach and may be some indication of the talent of the prober. It isn't clear to me what the 3 minute gap is between the first and second probe. This could be waiting for a response before running a script as the subsequent probes are pretty fast.

Active Targeting: Yes
Intent: Probing selected well known ports
Technique: Random anomalous flag sets
Severity: (C4+L1)-(CM4+N1)= 0 Low. Probably a script kiddie with limited skills. No evidence of system compromise.
Source: BlackIce Detect

Igor was one of the top students in the IDIC and his analysis pretty much says it all. This is one example of a number of violations of TCP state used for scanning. These are often referred to as half open probes. This term while not technically precise is used to describe probes that pretend to be past step one, the SYN or active open in a TCP 3-way handshake.

ACK SCAN
Submitted by Igor Gashinsky - http://www.sans.org/y2k/practical/Igor_Gashinsky.doc
Most traffic has been gathered using Shadow IDS.

```
04:29:37.820000 scanner.com > A.B.C.2: icmp: echo request
…
05:04:16.452203 scanner.com.51031 > A.B.C.2.1540: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.374: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.558: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.472: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.393: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.151: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.130: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.72: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.1005: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.22273: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.551: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.2108: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.908: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.440: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.305: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.1356: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.681: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.674: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.308: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.850: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.956: . 3686755777:3686755777(0) ack 0 win 2048
05:04:16.452203 scanner.com.51031 > A.B.C.2.414: . 3686755777:3686755777(0) ack 0 win 2048
…
```

Active Targeting	**YES**
History	**None previous**
Technique	A very fast ACK scan of the Honeypot machine
Analysis	This appears to be a pure ACK scan targeting a Honeypot. The lone ping in the beginning, and the speed of the scan indicate a script, quite possibly the new 2.30BETA17 version of NMAP (the first to offer ACK scanning). This scanning technique is designed to be stealthy, and would penetrate most non-state-aware firewall implementation. It is intended for mapping the firewall rule-bases, by waiting for either a RST, and ICMP Unreachable, or nothing to come back from the port, and based on that determine the rulebase (more information could be found at http://www.insecure.org/nmap/index.html#new) . This indicates that the attacker is using the latest tools, and since he is scanning the Honeypot, is not familiar with the network.
Threat	**MEDIUM**, the attacker is probing the defenses of the Honeypot, in preparation for an attack.

Smurf attacks rely on poorly configured sites that do not have proper ICMP filtering. These sites are known as amplifiers. If my.dot.com was a smurf amplifier it would return a large number of echo replies to the target, pitt.edu. www.netscan.org has a list of smurf amplifier sites, if your site is on the list, try to get that fixed. As the analysis implies, Pitt.edu is almost certainly spoofed and is the intended victim in this attack.

SMURF ATTACK
Submitted by Carolyn Willey - http://www.sans.org/y2k/practical/Carolyn_Willey.doc

136.142.78.53 > my.dot.com.255
00:00:42.380000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:01:20.250000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:01:58.120000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:02:35.980000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:03:13.850000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:03:51.710000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:04:29.580000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:05:07.450000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:05:45.340000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:06:23.200000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:07:01.090000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:07:38.930000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:08:16.830000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:08:54.690000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:09:32.560000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:10:10.420000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)
00:10:48.300000 xxx.xxx.pitt.edu > my.dot.com.255: icmp: echo request (DF)

This was an interesting trace because it appeared in the hourly wrap up for every hour of the three days that I analyzed (as well as before and after those three days, upon further investigation).

Evidence of active targeting? Definitely. ICMP echo requests very regularly and for a sustained time period from this (spoofed) site.

Techniques? The offending site (not pitt.edu) was sending frequent icmp echo requests to our broadcast (.255) address. Upon further investigation, I realized (from this and other hourly wrap-ups of the same trace) that we were getting this pattern exactly once every 38 seconds. This attack was obviously scripted, for it to happen for that long and with that precise frequency.

History? This attack appeared consistently over a number of days.

Analysis - A "smurf" denial of service attack aimed at pitt.edu, with our site being used as the middleman. I found it very interesting that the attack was sustained for such a long period of time.

One the hardest things to detect and analyze are probes that are low and slow and hidden in the noise of "normal" or even abnormal traffic. Jim Edwards, GCIA, has provided a trace that shows the analysis process he used to find signal from the noise.

LOW AND SLOW PROBE
Submitted by Jim Edwards - http://www.sans.org/y2k/practical/Jim_Edwards.doc

```
(sort 1)
LOG_DATE LOG_TIME ACTION PROTO SRC DST SERVICE S_PORT LEN
04-APR-2000 00:54:58 drop tcp etranz1.flyingcroc.net        my-net.107.84     9853  10943
40
31-MAR-2000 02:43:42 drop tcp 209.247.108.212               my-net.107.9      4312  11230
40
04-APR-2000 19:33:31 drop tcp 210.92.121.162                my-net.107.1      55775 11949
40
04-APR-2000 21:23:40 drop tcp 210.92.121.162                my-net.107.1      55775 11949
40
02-APR-2000 03:25:08 drop tcp 208.51.159.10                 my-net.107.14     18538 12033
40
31-MAR-2000 01:50:16 drop tcp netcom5.netcom.com            my-net.107.53     61549 12084
40
04-APR-2000 09:28:28 drop tcp dialup-unix.rapidenet.net     my-net.107.149    8137  12545
40
03-APR-2000 20:43:41 drop tcp 200.192.57.42                 my-net.107.124    41207 133
      40
04-APR-2000 18:12:12 drop tcp monet.telebyte.nl             my-net.107.192    58757 13543
40
30-MAR-2000 19:47:10 drop tcp 216.181.39.5                  my-net.107.89     63414 13678
40
03-APR-2000 22:32:26 drop tcp ahhhh.uhhhhhh.uhhhhh.ahhhhhh.com my-net.107.89  63411 13678
40
02-APR-2000 03:31:41 drop tcp 216.33.187.17                 my-net.107.245    3811  16163
40
31-MAR-2000 16:45:40 drop tcp 209.163.42.235                my-net.107.245    6371  16163
40
02-APR-2000 00:23:33 drop tcp elite.cyberphreak.net         my-net.107.21     8724  16344
40
30-MAR-2000 15:49:42 drop tcp atlanta.ga.us.undernet.org    my-net.107.246    19021 16994
40
30-MAR-2000 18:55:49 drop tcp 211.40.177.75                 my-net.107.246    53581 16995
40
31-MAR-2000 11:08:27 drop tcp w1.yahoo.com                  my-net.107.65     55226 17478
40
01-APR-2000 17:16:35 drop tcp sexhead.org                   my-net.107.38     43369 19289
40
02-APR-2000 12:59:05 drop tcp is.secksi.net                 my-net.107.73     62811 20025
40
02-APR-2000 16:16:37 drop tcp webqual.com                   my-net.107.83     57356 20911
40

(sort 2)
LOG_DATE LOG_TIME ACTION PROTO SRC DST SERVICE S_PORT LEN
01-APR-2000 17:11:01 drop tcp sexhead.org my-net.106.29  41132 6134  40
```

```
01-APR-2000 17:13:10 drop tcp sexhead.org my-net.104.29  27757 60852 40
01-APR-2000 17:15:37 drop tcp sexhead.org my-net.106.51  63093 62818 40
01-APR-2000 17:16:35 drop tcp sexhead.org my-net.107.38  43369 19289 40
01-APR-2000 17:24:06 drop tcp sexhead.org my-net.106.11  6213  44373 40
01-APR-2000 17:24:15 drop tcp sexhead.org my-net.105.119 18516 28134 40
01-APR-2000 17:27:28 drop tcp sexhead.org my-net.104.85  22673 20375 40
02-APR-2000 18:01:32 drop tcp sexhead.org my-net.104.111 32550 30767 40
02-APR-2000 18:01:44 drop tcp sexhead.org my-net.107.70  26914 31510 40
02-APR-2000 18:07:35 drop tcp sexhead.org my-net.104.20  64422 62339 40
02-APR-2000 18:28:50 drop tcp sexhead.org my-net.107.37  9302  42953 40
02-APR-2000 18:30:54 drop tcp sexhead.org my-net.107.94  19619 24584 40
02-APR-2000 18:32:11 drop tcp sexhead.org my-net.107.29  48771 39404 40
02-APR-2000 18:34:20 drop tcp sexhead.org my-net.107.7   59582 30449 40
```

Active Targeting: Yes

History: This started on 31 Mar 2000 and has been going on for several days so far.

Technique: The actual attackers address seem to be hidden within a large amount of spoofed addresses. They are using a low and slow scripted scan to try to probe our network. See comments below for more info on how I came to this conclusion and why I believe this is a directed scan and not just spurious traffic. The traffic I included is general traffic to all my network and specific traffic coming from one of the spoofed addresses.

Intent: This seems to be a scan of some kind although I have not been able to find any commonly scanned ports or trojans.

Impact: The impact is negligible since the firewall blocked all the packets.

Comments: This one is kind of iffy and took me several hours to work out. I had started seeing a lot of spurious traffic to an inactive network. We have one Class C (ny-net.107) that we use for small subnets and special networks, almost all of which are totally in-house and do not connect to the Internet. No PC has ever been assigned these IP addresses. When traffic started showing up, it raised a red flag and then I saw traffic from sexhead.org. Being in an agency that is sensitive to such activity, I immediately checked to see if any outgoing traffic to this site was present and it was not. I even checked all outgoing traffic to the network it belongs to and none showed up. Since we put all our firewall logs in an Oracle database every night, sorting and searching is very easy so I pulled all the traffic from sexhead and found exactly seven packets on two different days, all with non-existent ports, all exactly 40 bytes in length (see sort 2). There was also the fact that the .29 address to 3 different networks showed up which made me think more that this was not just spurious traffic. My first conclusion was that this machine was scanning us for existing networks. While I was working this, however, I noticed several other bits of similar traffic, once again to this non-existent network as well as my active networks. The more I looked the more I found, all seemingly starting on Mar 31st, there was very little of this kind of traffic before then. I eventually was able to sort out some of the activity to the 107 network and sorted it by source port and found something very interesting. There was some traffic to the same IP address, using the same source port but from completely different source addresses on different days. The chances of this happening randomly to a non-existent network is nearly impossible (see sort 1). There were also a few records that matched exactly except for the time. It is very difficult to separate the wheat from the chaff in this case so some of this traffic may not be a part of this scan but what little evidence I have leads me to believe that this is a direct attempt at network mapping or

searching for some trojan. I have, unfortunately, not been able to determine exactly which since neither the destination or source ports match to anything known.

Here is another example of scanning for NetBIOS Name Service. Note there is no way to conclude this is a chode virus from the pattern below, but this a fine example of analysis to determine someone is using address spoofing to help cover their tracks. The use of information like TTLs for traffic analysis is strongly encouraged in intrusion detection courses.

ADDRESS SPOOFING
Submitted by James Dean Kirby - http://www.sans.org/y2k/practical/James_Dean_Kirby.doc
Source of logs = Shadow system

```
07:38:52.010523 169.254.94.104.137 > OurNet.2.137: udp 50 (ttl 118, id 46322)
07:38:52.027246 207.172.37.147.137 > OurNet.2.137: udp 50 (ttl 118, id 46578)
07:38:53.498651 169.254.94.104.137 > OurNet.2.137: udp 50 (ttl 118, id 46834)
07:38:53.507509 207.172.37.147.137 > OurNet.2.137: udp 50 (ttl 118, id 47090)
07:38:54.997116 169.254.94.104.137 > OurNet.2.137: udp 50 (ttl 118, id 47346)
07:38:55.027855 207.172.37.147.137 > OurNet.2.137: udp 50 (ttl 118, id 47602)
07:39:02.695115 169.254.94.104.137 > OurNet.3.137: udp 50 (ttl 118, id 48882)
07:39:02.724882 207.172.37.147.137 > OurNet.3.137: udp 50 (ttl 118, id 49138)
07:39:07.455380 169.254.94.104.137 > OurNet.3.137: udp 50 (ttl 118, id 49394)
07:39:07.464581 207.172.37.147.137 > OurNet.3.137: udp 50 (ttl 118, id 49650)
07:39:08.944226 169.254.94.104.137 > OurNet.3.137: udp 50 (ttl 118, id 49906)
07:39:08.974544 207.172.37.147.137 > OurNet.3.137: udp 50 (ttl 118, id 50162)
07:39:36.907120 169.254.94.104.137 > OurNet.5.137: udp 50 (ttl 118, id 54514)
07:39:36.915327 207.172.37.147.137 > OurNet.5.137: udp 50 (ttl 118, id 54770)
07:39:38.404332 169.254.94.104.137 > OurNet.5.137: udp 50 (ttl 118, id 55026)
07:39:38.434785 207.172.37.147.137 > OurNet.5.137: udp 50 (ttl 118, id 55282)
07:39:39.905351 169.254.94.104.137 > OurNet.5.137: udp 50 (ttl 118, id 55538)
07:39:39.914171 207.172.37.147.137 > OurNet.5.137: udp 50 (ttl 118, id 55794)
…
```

Existence: 2 IP address appearing to come from opposite coasts one being Erols (which I think is East Coast only) and the U of Southern California at Marina Del Rey. Obviously one, if not both, of the address is spoofed.

History: These connect attempts went on to scan each of our public addresses on that subnet.

Techniques: Basic script-kiddie automated netBios scan looking for open NetBios hosts.

Intent: 911/Chode virus looking for hosts? Definitely scanning for open hosts. Possibly attempting DoS on the 2 source addresses.

Targeting: Scary. Targets NOT each address in the domain, NOT each address that is resolvable, NOT each address that responds to pings, but each address that actually carries data. This shows an extreme knowledge of our network.

Severity: (4+4) – (2+5) = 1

Analysis: This scan falls right in with the 137-137 being reported on GIAC these past days. At least one of these IP addresses is spoofed. This is evident not only from the identical TTLs but the packet ID's and time stamps indicate that the attacker activated 2 scripts simultaneously (or 2 threads in the same attack program) with each instance sending 3 crafted UDP packets at each host. While this does not fit the 5 packets reported at GIAC, it is perhaps a different signature of the same worm.

Another example of looking beyond the port number. Most analysts will see a packet to 1234 and classify it as a Trojan and move to the next case. Jerry dug deeper and in doing so increases all of our understanding.

NOVELL ON TROJAN PORT
Submitted by Jerry Shenk - http://www.sans.org/y2k/practical/Jerry_Shenk.doc

Detect 3

In this sequence of packets, both addresses are Netware servers. 10.94.3.1 is running Netware 5.0 and 10.1.1.8 is running Netware 4.

This traffic bears further review because of the destination port 1234 on udp. This looks like it could be an Ultors Trojan back door 10.94.3.1 and 10.1.1.8 are both Netware servers. I then collected a large sequence of all traffic between these two hosts. This sample shows no traffic other than the udp traffic between port 524 on the Netware 5 server and port 1234 on the Netware 4. This revealed that there was also tcp traffic between port 4326 on 10.1.1.8 and port 524 on 10.94.3.1. I am suspecting that this is not in fact a trojan because of the computers involved so I went to the Novell web site and found a document on the IP ports used in Netware 5 (TID#**10013531)**. It turns out that Novell servers do NCP (Netware Core Protocol) communication over tcp 524 and time syncronization over udp 524. The use of port 1234 seems to simply be the ephemeral port that was next in line for use.

This network has 25-30 Netware servers so this will be a difficult traffic to eliminate from the IDS without allowing a hole for the Ultors Trojan. Given the fact that this network has all private addresses on the inside I think it's safe to eliminate the port 1234 alarms for the internal sensors but we can still allow it on the external sensor because this traffic still should not be being sent out to the internet.

Initial Packet Sequence (collected using snort)

```
04/05-21:43:00.471844 0:90:27:54:B5:D -> 0:10:4B:97:F:40 type:0x800 len:0x3C
10.94.3.1:524 -> 10.1.1.8:1234 UDP TTL:128 TOS:0x0 ID:62921
Len: 22
3E 3E 00 62 00 00 00 00 62 3F 00 00 00 00 00 00  >>.b....b?......
51 00                                            Q.

04/05-21:43:00.471844 0:10:4B:97:F:40 -> 0:A0:C9:FB:53:BB type:0x800 len:0x3C
10.94.3.1:524 -> 10.1.1.8:1234 UDP TTL:127 TOS:0x0 ID:62921
Len: 22
3E 3E 00 62 00 00 00 00 62 3F 00 00 00 00 4B 4B  >>.b....b?....KK
4B 4B                                            KK

04/05-21:49:03.811844 0:90:27:54:B5:D -> 0:10:4B:97:F:40 type:0x800 len:0x3C
10.94.3.1:524 -> 10.1.1.8:1234 UDP TTL:128 TOS:0x0 ID:8138
Len: 22
3E 3E 00 62 00 00 00 00 62 3F 00 00 00 00 63 50  >>.b....b?....cP
00 00                                            ..

04/05-21:49:03.811844 0:10:4B:97:F:40 -> 0:A0:C9:FB:53:BB type:0x800 len:0x3C
10.94.3.1:524 -> 10.1.1.8:1234 UDP TTL:127 TOS:0x0 ID:8138
Len: 22
3E 3E 00 62 00 00 00 00 62 3F 00 00 00 00 17 17  >>.b....b?......
17 17                                            ..
```

2nd Packet Sequence (collected using tcpdump)

```
Wed Apr  5 21:54:52 EDT 2000
```

```
tcpdump: listening on eth0
21:55:07.141844 10.94.3.1.524 > 10.1.1.8.1234: udp 14
21:55:07.141844 10.94.3.1.524 > 10.1.1.8.1234: udp 14
21:55:07.181844 10.1.1.8.1234 > 10.94.3.1.524: udp 14
21:55:07.181844 10.1.1.8.1234 > 10.94.3.1.524: udp 14
21:55:39.911844 10.1.1.8.4326 > 10.94.3.1.524: P 915600012:915600268(256) ack 1972944771 win 65424 (DF)
21:55:39.921844 10.1.1.8.4326 > 10.94.3.1.524: P 0:256(256) ack 1 win 65424 (DF)
21:55:39.921844 10.94.3.1.524 > 10.1.1.8.4326: P 1:207(206) ack 256 win 20496 (DF)
21:55:39.921844 10.94.3.1.524 > 10.1.1.8.4326: P 1:207(206) ack 256 win 20496 (DF)
21:55:40.001844 10.1.1.8.4326 > 10.94.3.1.524: P 256:372(116) ack 207 win 65218 (DF)
21:55:40.001844 10.1.1.8.4326 > 10.94.3.1.524: P 256:372(116) ack 207 win 65218 (DF)
21:55:40.001844 10.94.3.1.524 > 10.1.1.8.4326: P 207:397(190) ack 372 win 20380 (DF)
21:55:40.001844 10.94.3.1.524 > 10.1.1.8.4326: P 207:397(190) ack 372 win 20380 (DF)
21:55:40.091844 10.1.1.8.4326 > 10.94.3.1.524: . ack 397 win 65028 (DF)
21:55:40.091844 10.1.1.8.4326 > 10.94.3.1.524: . ack 397 win 65028 (DF)
22:00:23.041844 10.1.1.8.524 > 10.94.3.1.4105: . 918296195:918296196(1) ack 3820007976 win 22404 (DF)
22:00:23.041844 10.1.1.8.524 > 10.94.3.1.4105: . 0:1(1) ack 1 win 22404 (DF)
22:00:23.041844 10.94.3.1.4105 > 10.1.1.8.524: . ack 1 win 65535 (DF)
22:00:23.041844 10.94.3.1.4105 > 10.1.1.8.524: . ack 1 win 65535 (DF)
22:01:10.481844 10.94.3.1.524 > 10.1.1.8.1234: udp 14
```

3rd Packet Sequence (collected using tcpdump w/-w option and then using snort to view)

```
04/05-22:05:06.371844 0:A0:C9:FB:53:BB -> 0:10:4B:97:F:40 type:0x800 len:0x6E
10.1.1.8:427 -> 10.94.3.1:427 UDP TTL:32 TOS:0x0 ID:1914
Len: 75
01 02 00 43 00 00 65 6E 00 6A 00 25 00 00 00 01  ...C..en.j.%....
00 00 00 2F 73 65 72 76 69 63                    .../servic
```

…..much similar traffic deleted to shorten the report

```
04/05-22:05:58.471844 0:10:4B:97:F:40 -> 0:90:27:54:B5:D type:0x800 len:0x21A
10.1.1.8:427 -> 10.94.3.1:427 UDP TTL:31 TOS:0x0 ID:1922
Len: 504
01 07 01 F0 00 00 65 6E 00 6A 00 8D 00 00 01 E0  ......en.j......
28 73 76 63 6E 61 6D 65 2D 77                    (svcname-w

04/05-22:06:07.241844 0:90:27:54:B5:D -> 0:10:4B:97:F:40 type:0x800 len:0x136
10.94.3.1:4105 -> 10.1.1.8:524 TCP TTL:128 TOS:0x0 ID:38275  DF
*****PA* Seq: 0xE3B0B228   Ack: 0x36BC1684   Win: 0xFFFF
44 6D 64 54 00 00 01 00 00 00 00 01 00 00        DmdT..........

04/05-22:06:07.241844 0:10:4B:97:F:40 -> 0:A0:C9:FB:53:BB type:0x800 len:0x136
10.94.3.1:4105 -> 10.1.1.8:524 TCP TTL:127 TOS:0x0 ID:38275  DF
*****PA* Seq: 0xE3B0B228   Ack: 0x36BC1684   Win: 0xFFFF
44 6D 64 54 00 00 01 00 00 00 00 01 00 00        DmdT..........
```

…..much similar traffic deleted to shorten the report

```
04/05-22:06:47.931844 0:10:4B:97:F:40 -> 0:90:27:54:B5:D type:0x800 len:0x3C
10.1.1.8:4326 -> 10.94.3.1:524 TCP TTL:127 TOS:0x0 ID:1933  DF
******A* Seq: 0x3692FD04   Ack: 0x7598C60B   Win: 0xFE02
00 00 00 00 00 00                                ......

04/05-22:07:13.801844 0:90:27:54:B5:D -> 0:10:4B:97:F:40 type:0x800 len:0x3C
10.94.3.1:524 -> 10.1.1.8:1234 UDP TTL:128 TOS:0x0 ID:40248
Len: 22
```

3E 3E 00 62 00 00 00 00 62 3F 00 00 00 00 63 50 >>.b....b?....cP
00 00   ..

…..much similar traffic deleted to shorten the report

We teach the demon net silly TCP syndrome in our courses since these patterns are so widespread and have been going on for so long. We also teach students to be aware there are other sources of broken packets that have nothing to do with (allegedly) bad hardware. This trace is instructive as a classic example of the madness of the Demon Internet's routers.

ILLEGAL FLAG COMBINATIONS
Submitted by Paul Sears - http://www.sans.org/y2k/practical/Paul_Sears.doc
These detects are from SHADOW logs that were analyzed from March 31, 2000 through April 07, 2000. Interesting events were extracted from the logs and discussed in this document

Detect 1 – Illegal flag combinations from Demon.net

```
Date: Apr03 - PDT: 08:00
194.217.242.89 > 192.168.1.3
08:30:22.682933 anchor-post-31.mail.demon.net.31510 > proxy.mysite.com.31501: SFR
8954981:8956445(1464) ack 2065694720 win 0 urg 4096 (DF)

194.217.242.91 > 192.168.1.3
08:22:10.134217 anchor-post-33.mail.demon.net.27045 > proxy.mysite.com.27005: SRP
10944190:10945650(1460) win 128 urg 25280 (DF)
08:26:30.577838 anchor-post-33.mail.demon.net.27035 > proxy.mysite.com.27005: SFRP
11519773:11521245(1472) win 0 urg 32768 (DF)

Date: Apr05 - PDT: 10:00
194.217.242.38 > 192.168.1.3
10:25:13.720179 finch-post-10.mail.demon.net.7766 > proxy.mysite.com.1077: SFR
2857770:2859210(1440) win 98 <[bad opt]> (DF)
```

Description of detect
These packets were flagged as having illegal TCP flag combinations.

Active Targeting
No, normal email exchange.

Intent
Email exchange between Demon.net and mysite.com. No malicious intent stated by Demon.net.

History
Demon.net has stated they have a faulty router that creates packets with these illegal flag combinations.

Analysis
Syn-Reset-Push and Syn-Fin-Reset-Push are not legal combinations of TCP flags. It is actually kind of cool to see this after hearing about it at SANS2000. Of course, Demon.net really should fix that silly router...

Threat Level: ■ ■ ■ ■
Criticality = 3
Lethality = 1
System Countermeasures = 4
Network Countermeasures = 4
Severity =(3+1) - (4+4) = -4

Active and available echo ports can be used as amplifiers in a large number of attacks. Analysts should be alert for echo either from source or destination.

UDP SCAN
Submitted by Lee Brandt - http://www.sans.org/y2k/practical/Lee_Brandt.doc

-source-	-dest-	-sport-	-dport-	-protocol-
212.187.65.86	203.5.67.63	7744	7	17
212.187.65.86	205.5.66.128	6537	7	17
212.187.65.86	205.5.66.63	29432	7	17
212.187.65.86	205.5.66.128	15793	7	17
212.187.65.86	205.5.66.191	17367	7	17
212.187.65.86	205.5.67.63	29210	7	17
212.187.65.86	205.5.67.127	351	7	17
212.187.65.86	205.5.66.127	17330	7	17

Source of Trace: Joe@ITS.UNIMELB.EDU.AU sent this trace to the Security Focus Incident Reporting System.

Active Targeting: Yes

History: Unknown – access to previous logs was not available.

Technique: The Destination port is constant (7 - echo).
The source address appears to be spoofed
The destination address are broadcast addresses.
UDP scan
Single Service Port Scanned

Intent: Malicious in nature. Looking for a vulnerable host to launch a DOS Attack.

Analysis: This appears to be fraggle attack. The attacker is looking for a victims to launch a larger Denial of Service (DdoS) against someone. Again, this could be the a recon for a possible DoS against this network.

Severity of Attack: Medium

Component	Score	Comments
Criticality	4	Attack is not focusing on a single host.
Lethality	4	The network could subject to a DoS attack
System Countermeasures	4	Modern operating system, can not tell if all the patches were installed.
Network Countermeasures	2	The network firewall has been permissive
Severity Score	2	Severity = (Criticality + Lethality) – (system countermeasures + net countermeasures)

This trace illustrates a common trap analysts fall into. It is rare, therefore it must be malicious. In the example below, I can't see anything wrong with the ICMP packet, it isn't big enough for denial of service, the offsets line up correctly, the MF (More Fragment) is set on all fragments except the last. All of the offsets are divisible by 8. So it is NOT denial of service. However, the detect is still interesting, you do not see pings this big very often. The question then is why.

ATTEMPTED DENIAL OF SERVICE ATTACK
Submitted by Tony Adams - http://www.sans.org/y2k/practical/Tony_Adams.doc

04-Apr-00 13:52:08.767368 38.222.85.2 > A.B.C.8: icmp: echo request (frag 42:512@0+)
04-Apr-00 13:52:08.768019 38.222.85.2 > A.B.C.8: (frag 42:512@512+)
04-Apr-00 13:52:08.768549 38.222.85.2 > A.B.C.8: (frag 42:512@1024+)
04-Apr-00 13:52:08.768980 38.222.85.2 > A.B.C.8: (frag 42:512@1536+)
04-Apr-00 13:52:08.769275 38.222.85.2 > A.B.C.8: (frag 42:512@2048+)
04-Apr-00 13:52:08.769504 38.222.85.2 > A.B.C.8: (frag 42:512@2560+)
04-Apr-00 13:52:08.769732 38.222.85.2 > A.B.C.8: (frag 42:512@3072+)
04-Apr-00 13:52:08.769959 38.222.85.2 > A.B.C.8: (frag 42:512@3584+)
04-Apr-00 13:52:08.770411 38.222.85.2 > A.B.C.8: (frag 42:512@4096)
04-Apr-00 13:52:08.779000 38.222.85.2 > A.B.C.8: icmp: echo request

Trace Information: This trace was captured on our DMZ.
Active Targeting: Yes
History: This IP address did a scan of our IP address range for host with open port 111 (portmapper) 24 hours earlier.
Technique: Direct attempt at a denial of service attack, by sending fragmented ping packets.
Analysis: This is a deliberate attempt to take this machine offline. The attacker is sending fragmented ping packets, which some TCP stacks can not properly handle. The attack can cause a machine to hang thus the denial of service. This could be a form of the "Ping of Death". The IP address being used belongs to PSI.net, probably part of a dial-in pool.
Threat: This would constitute a HIGH threat due to the deliberate attempt to take the system offline.

This trace shows the importance of watching outbound traffic with the same intensity as we examine inbound. Kevin Peterson, GCIA has been very kind, I would have been rather upset if I had outbound traffic to hardcoded addresses leaving my network.

FALSE POSITIVE
Submitted by Kevin Peterson - http://www.sans.org/y2k/practical/Kevin_Peterson.doc

09:26:59.450743 0:e0:16:98:ed:85 8:0:20:a8:73:16 ip 88: htp-0b.bbt.com.snmp > w.x.54.5.snmp-trap: |30|2c|02|01|04|09C=SNMP_trap |a4|1cTrap(28) |06|09E:1770.3.1 |40|04[w.x.54.209]|02|01 coldStart|02|01 |43|010|30|00 (ttl 63, id 17090)
4500 004a 42c2 0000 3f11 2957 c09a 5883 E..JB...?.)W..X.
c067 3605 00a1 00a2 0036 0000 302c 0201 .g6.....6..0,..
0004 0953 4e4d 505f 7472 6170 a41c 0609 ...SNMP_trap....
2b06 0104 018d 6a03 0140 04c0 6736 d102 +.....j..@..g6..
0100 0201 0043 0100 3000 ....C..0.
09:26:59.453618 0:e0:16:98:ed:85 8:0:20:a8:73:16 ip 111: htp-0b.bbt.com.snmp > w.x.54.5.snmp-trap: |30|43|02|01[version(1)!=0] (ttl 63, id 17091)
4500 0061 42c3 0000 3f11 293f c09a 5883 E..aB...?.)?..X.
c067 3605 00a1 00a2 004d 0000 3043 0201 .g6.....M..0C..
0104 0953 4e4d 505f 7472 6170 a733 0201 ...SNMP_trap.3..
0002 0100 0201 0030 2830 0d06 082b 0601 ....0(0...+..
0201 0103 0043 0100 3017 060a 2b06 0106 .....C..0...+...
0301 0104 0100 0609 2b06 0106 0301 0105 ....+.......
01.. ....

ANALYSIS

Active Targeting
Yes. It was coming from mutliple sources from inside our network.
Intent
It ended up being misconfigured software. It could have been an attempt to gather information from the inside, similar to a trojan sending out information.
Technique
Used snmp traps.
History
These packets and many others like it were picked up after seeing log messages in our firewall about an inside machine trying to send snmp trap messages to a machine out on the internet. After investigating the destination address, I found that it belonged to a company that happened to be a supplier of some snmp code that we were using. After talking to the developer, he indicated that there were some default addresses in the code that he did not change because he didn't know what to change them to. I suggested 127.0.0.1.
Severity
Low(-1). The snmp traps were being stopped at the firewall so the information was not getting out into the wild.

IMAP accounts for a large number of the compromises that have occurred on Unix systems between 1997 and 2000. While it is a bit less of a problem today, it is still a major scan. Analysts are advised to never assume they aren't running IMAP. If you can get permission, run a tool like nmap on your internal network from time to time so you have a good idea of what your active services are.

IMAP
Submitted by Kirk Becker - http://www.sans.org/y2k/practical/Kirk_Becker.txt
Source of logs: via our E-mail server at work. It is equipped with TCP Wrappers and PortSentry to help defend itself.

```
Mar 16 22:36:57 mail imapd[26325]: connect from 202.102.12.10
Mar 16 22:36:57 mail imapd[26325]: imap service init from 202.102.12.10
Mar 16 22:37:24 mail imapd[26325]: command stream end of file, while reading
line user=??? host=[202.102.12.10]

Analysis:
Attempting to connect to imap, which we are running.  The host is an ISP in
China.  We have seen about 3 different attempted connects like this from Asia
(once from Korea and the other from Singapore).

Severity of Incident: Probably total to about a 4.
Criticality - E-mail server - 4
Lethality - Could be an imap exploit, I doubt it is a "wrong number" from China
to a small organization in Florida - 4, maybe even a 5.
System - Same E-mail server as before, TCP Wrappers and PortSentry, but it
allowed the connect - 3
Firewall - NAT to the mail server - 2
```

Checking for back doors is hard, but necessary. Many defenders rely too heavily on their firewalls, but if there are back doors it is possible to attack systems by source routing though backdoors. Here is the story of an analyst finding a backdoor. I have also seen a pattern similar to this when employees have a corporate laptop at work and connect to the Internet at home. You can often see exactly what they are doing since they still have the organization's DNS server as their default. This can be very educational!

LET'S GET AROUND THE INTERNAL FIREWALL...
Submitted by John Eisenhauer - http://www.sans.org/y2k/practical/john_eisenhauer.doc

This also showed up when testing the same windump filter as above. We were looking for 'address leakage' again. This time we found some.

23:13:04.379538 dhcp-43.cablemodem.isp.net.137 > ns2.isp.net.53: 32+ A? USERDOMAIN.isp.net. (47)
23:13:04.379891 dhcp-43.cablemodem.isp.net.137 > ns2.isp.net.53: 38+ A? USERDOMAIN.isp.net. (47)
23:13:05.879477 dhcp-43.cablemodem.isp.net.137 > ns3.isp.net.53: 26+ A? USERDOMAIN.isp.net. (47)
23:13:05.880187 dhcp-43.cablemodem.isp.net.137 > ns3.isp.net.53: 32+ A? USERDOMAIN.isp.net. (47)
23:13:05.880233 dhcp-43.cablemodem.isp.net.137 > ns2.isp.net.53: 38+ A? USERDOMAIN.isp.net. (47)
23:13:07.276172 dhcp-43.cablemodem.isp.net.137 > ns3.isp.net.53: 32+ A? USERDOMAIN.isp.net. (47)
23:13:07.360750 dhcp-43.cablemodem.isp.net.137 > ns2.isp.net.53: 38+ A? USERDOMAIN.isp.net. (47)
23:13:08.878850 dhcp-43.cablemodem.isp.net.137 > ns3.isp.net.53: 32+ A? USERDOMAIN.isp.net. (47)

Active Targeting: Yes
History: We have suspected for some time that dial-in users may be connected to the Internet via cable modem while concurrently connecting to our network. The implications of this are very bad. I will not go into that aspect of things. We now through our network of sensors have some visibility to this. Matching dial-in logs to the start and stop times of the traffic pointed to one particular user. We have suspected them in the past as being an offender in this area. A manager will talk to them again.
Technique: An uninformed user purchases a cable modem for home, connects their home system to it, and then dials into our corporate network. This effectively opens a backdoor into our network from the Internet.
Intent: The intent would be to circumvent the monitoring and access control that we have on our firewall.
Evaluation: These new 'always on' Internet connections are such a headache. Users want fast Internet access at home, and go out and get a cable modem. Regardless of intent, or whether or not the user knows what they are doing, they then dial into our network with their cable modem connected PC. This makes their unprotected PC a back door into our private network. A policy has been written to ban this, but we now need to find ways to identify the problem when it occurs.

In this case, when the Windows box connected via RAS, it's default gateway changed to our network. For whatever reason, it then began to send out traffic destined for its ISP through our network. This is what we see in the trace.
Severity: Criticality = 5 Core hosts could be exposed.
Lethality = 3 An attacker exploiting this potential back door could gain user access on the network at least.
System CM = 2 This particular kind of thing can expose 'weak' systems to common attacks.
Network CM = 1 The hope we have here is that our detection system catches this early on. The firewall cannot help with this.
Overall Severity = (5 + 3) – (2 + 1) = 5

We have talked about TCP fingerprinting or stack analysis. Note that SYN/FIN is part of this pattern. Queso is generally considered to be less advanced than nmap for this purpose.

QUESO SCAN
Submitted by Charles McCants - http://www.sans.org/y2k/practical/Charles-McCants.doc

```
20:29:20.332978 badguy.7721 > victim.www: S 606281182:606281182
(0) win 4660 (ttl 240, id 39058)
20:29:20.333140 victim.www > badguy.7721: S 2092188174:20921881
74(0) ack 606281183 win 32736 <mss 536> (ttl 64, id 17381)
20:29:20.340290 badguy.7722 > victim.www: S 606281182:606281182
(0) ack 0 win 4660 (ttl 240, id 39059)
20:29:20.340378 victim.www > badguy.7722: R 0:0(0) win 0 (ttl 2
55, id 17382)
20:29:20.349799 badguy.7723 > victim.www: F 606281182:606281182
(0) win 4660 (ttl 240, id 39060)
20:29:20.358508 badguy.7724 > victim.www: F 606281182:606281182
(0) ack 0 win 4660 (ttl 240, id 39061)
20:29:20.358629 victim.www > badguy.7724: R 0:0(0) win 0 (ttl 2
55, id 17383)
20:29:20.365636 badguy.7725 > victim.www: SF 606281182:60628118
2(0) win 4660 (ttl 240, id 39062)
20:29:20.365742 victim.www > badguy.7725: SF 2029898772:2029898
772(0) ack 606281183 win 32736 <mss 536> (ttl 64, id 17384)
20:29:20.373542 badguy.7726 > victim.www: P win 4660 (ttl 240,
id 39063)
20:29:20.385035 badguy.7727 > victim.www: S 606281182:606281182
(0) win 4660 (ttl 240, id 39064)
20:29:20.385138 victim.www > badguy.7727: S 4288947413:42889474
13(0) ack 606281183 win 32736 <mss 536> (ttl 64, id 17385)
20:29:20.451655 badguy.7721 > victim.www: R 606281183:606281183
(0) win 0 (ttl 49, id 34220)
20:29:20.484768 badguy.7725 > victim.www: R 606281183:606281183
(0) win 0 (ttl 49, id 55808)
20:29:20.529088 badguy.7727 > victim.www: R 606281183:606281183
(0) win 0 (ttl 49, id 45481)
```

Active Targeting?	Yes
Analysis	**Queso OS Detection Scan** By sending a specific sequence of malformed packets, queso can determine the target's operating system. This info can then be used to target specific attacks against the machine. See queso output below:
Intent	Reconn
Severity	(4+2)-(3+2)=1 Low

```
# /usr/local/sbin/queso -d victim
Starting badguy:7721 -> 38.254.242.30:80
IN  #0 : 80->7721 S:1 A:+1 W:7FE0 U:0 F: SYN ACK
IN  #1 : 80->7722 S:0 A: 0 W:0000 U:0 F: RST
IN  #3 : 80->7724 S:0 A: 0 W:0000 U:0 F: RST
IN  #4 : 80->7725 S:1 A:+1 W:7FE0 U:0 F: SYN FIN ACK
IN  #6 : 80->7727 S:1 A:+1 W:7FE0 U:0 F: SYN ACK
38.254.242.30:80        * Linux 2.0.35 to 2.0.9999 :)
```

The above output shows the return flags from the victim to the badguy. It's OS determination was correct.

This is another example of the Demon pattern. It is worth noting there are common Snort rules (as shown below) to detect most weird flag combinations.

INVALID TCP FLAGS
Submitted by Jonathan Good - http://www.sans.org/y2k/practical/Jonathan_Good.doc

This detect is from a snort analysis of a tcpdump capture
Initial Detect:

```
[**] Syn-Fin flags [**]
04/05-10:48:37.680454 195.173.97.31:769 -> x.y.z.22:43849
TCP TTL:52 TOS:0x0 ID:46652  DF
**SFR*A* Seq: 0x0   Ack: 0x4500002C   Win: 0x4000

[**] Null Scan - no flags [**]
04/05-10:51:49.802735 195.173.97.31:1528 -> x.y.z.22:4091
TCP TTL:52 TOS:0x0 ID:47029
******** Seq: 0x20EEDE   Ack: 0x100018   Win: 0x5197

[**] Reserved flags [**]
04/05-10:54:37.443644 195.173.97.31:104 -> x.y.z.22:15650
TCP TTL:52 TOS:0x0 ID:47554
2***R*AU Seq: 0x38352220   Ack: 0x68656967   Win: 0x3D22
```

Analysis: This was flagged due to illegal combinations of TCP flag bits.
There isn't active targeting because even though it looks like a very interesting scan, it is more likely a 'Demon' internet packet. The source is a web server (shop.starshiptitanic.com), hosted by The Digital Village, on a class C subnet from Demon.net. Review of firewall logs show a user connected to the web site prior to the glitch. Demon.net claims to have faulty Ascend routers that mangle some traffic.
Note: *Severity is based on initial detect only, as it would be used to determine how much analysis to devote to the detect.*

Severity		(Criticality		Lethality)		(System Countermeasures		Network Countermeasures)
	=	Firewall Targeted	+	Recon	-	Up to date Firewall patches	+	Restrictive Firewall
0		5		3		4		4

Severity after analysis: NA - not an active target

The next trace is a really pretty system trace submitted by intrusion researcher, Doug Steinbaum, GCIA. It shows the value of really scrubbing your system logs as well as the network intrusion detection events of interest. Over the long term, say 12 years email has been a favored and valuable target for attackers.

MAJORDOMO EXPLOIT
Submitted by Douglas Steinbaum - http://www.sans.org/y2k/practical/Douglas_Steinbaum.doc

Active Targeting:

Yes.

History:

No previous activity from the source address is known.

Analysis:

The log below was in an email message automatically sent by a Gauntlet firewall to its administrator. The log includes an email that the firewall refused to forward/deliver. The log shows that the mail message's REPLY-TO field included shell commands that, if executed, would attempt to telnet into one of our computers. Researching the source IP address revealed that it belonged to an individual in our organization's security department who had just installed a vulnerability scanner tool.

Intent

The individual who sent the mail to the firewall was attempting to exploit a majordomo list server vulnerability that allows arbitrary commands to be run as a privileged user. Several days later, the person who generated the suspicious traffic was identified as someone working in our security department. He had been scanning various hosts in our organization for vulnerabilities with the intention of alerting owners of such vulnerabilities so they could be fixed. No malice was intended. However, this illustrates the importance of alerting system administrators in one's organization of scanning activity so that they do not mistake activities of the security personnel as attacks and escalate them needlessly.

Log file:

```
   ----- The following addresses have delivery notifications -----
majordomo  (unrecoverable error)

   ----- Transcript of session follows -----
550 majordomo... User unknown

   ----- Original message follows -----

Return-Path: <nobody@localhost.mydomain.com>
Received: by firewall.mydomain.com; id JAA13170; Tue, 3 Aug 1999 09:16:14 -0400 (EDT)
Date: Tue, 3 Aug 1999 09:16:14 -0400 (EDT)
From: <nobody@localhost.mydomain.coml>
Message-Id: <199908031316.JAA13170@mydomain.com>
Received: from bad-d00d.mydomain.com (4.3.2.1) by firewall.mydomain.com via smap (4.1)
        id xma013162; Tue, 3 Aug 99 09:16:09 -0400
To: majordomo@localhost.mydomain.com
Subject: Message for you
Reply-To: XX~.`telnet\${IFS}1.2.3.4\${IFS}4878`.q~/ad=AA/c=CC\\@ZZ.box.mydomain.com

lists
```

Of course we had to have a NETBUS detect, but this one is particularly interesting since the analyst took the trouble to do significant analysis of the TCP traffic.

NETBUS
Submitted by Jim Clausing - http://www.sans.org/y2k/practical/Jim_Clausing.txt

```
A couple of people trolling for NetBus

   Desc:  Probes for the NetBus trojan

   Analysis: These folks are just trolling hoping to find a machine running
   netbus (known trojan that runs on port 12345).  In the first and
   third cases, the IP ID #'s increase by 256 indicating they are
   consecutive packets from the source machine.  They aren't particularly
   fast, but still clearly automated.  In the second case, the IP ID
   numbers differ by 1536 indicating 6 packets in 3 seconds, not fast
   but the machine is obviously doing something else as well.  In the
   fourth one, the machine is also doing something else, though not very
   heavily loaded.  The IP ID numbers differ by 512 (indicating another
   packet in between) and the 2 packets are 3 seconds apart.  Since the
   target is a Unix box, these probes are merely annoying.

   Severity: Low

Apr  7 21:25:52 gauss kernel: Packet log: input DENY ppp0 PROTO=6
24.29.76.142:1514 me.at.home.186:12345 L=48 S=0x00 I=6775 F=0x4000 T=112 SYN
(#26)
Apr  7 21:25:55 gauss kernel: Packet log: input DENY ppp0 PROTO=6
24.29.76.142:1514 me.at.home.186:12345 L=48 S=0x00 I=7031 F=0x4000 T=112 SYN
(#26)


Apr  7 21:26:47 gauss kernel: Packet log: input DENY ppp0 PROTO=6
209.90.222.153:1800 me.at.home.186:12345 L=44 S=0x00 I=55315 F=0x4000 T=111 SYN
(#26)
Apr  7 21:26:50 gauss kernel: Packet log: input DENY ppp0 PROTO=6
209.90.222.153:1800 me.at.home.186:12345 L=44 S=0x00 I=56851 F=0x4000 T=111 SYN
(#26)


04/07-22:19:47.635071 209.143.42.87:1584 ->  me.at.home.186:12345
TCP TTL:121 TOS:0x0 ID:18698  DF
**S***** Seq: 0x21C8F9   Ack: 0x0   Win: 0x2000
TCP Options =>  MSS: 536 NOP NOP SackOK

04/07-22:19:48.665048 209.143.42.87:1584 ->  me.at.home.186:12345
TCP TTL:121 TOS:0x0 ID:18954  DF
**S***** Seq: 0x21C8F9   Ack: 0x0   Win: 0x2000
TCP Options =>  MSS: 536 NOP NOP SackOK

04/07-22:19:50.605059 209.143.42.87:1584 ->  me.at.home.186:12345
TCP TTL:121 TOS:0x0 ID:19210  DF
**S***** Seq: 0x21C8F9   Ack: 0x0   Win: 0x2000
TCP Options =>  MSS: 536 NOP NOP SackOK
```

```
04/07-22:19:51.625055 209.143.42.87:1584 ->  me.at.home.186:12345
TCP TTL:121 TOS:0x0 ID:19466  DF
**S***** Seq: 0x21C8F9   Ack: 0x0   Win: 0x2000
TCP Options =>  MSS: 536 NOP NOP SackOK


04/07-22:23:00.535062 148.235.8.111:1816 ->  me.at.home.186:12345
TCP TTL:18 TOS:0x0 ID:59149  DF
**S***** Seq: 0x3498BD   Ack: 0x0   Win: 0x2000
TCP Options =>  MSS: 1460

04/07-22:23:03.935088 148.235.8.111:1816 ->  me.at.home.186:12345
TCP TTL:18 TOS:0x0 ID:59661  DF
**S***** Seq: 0x3498BD   Ack: 0x0   Win: 0x2000
TCP Options =>  MSS: 1460
```

This trace shows a fine example of network mapping to support load balancing. This is a constantly changing pattern as ISPs keep trying to develop combinations that don't anger system administrators. You can imagine that some sites do not take kindly to folks probing their DNS ports.

COLLECT RTT FIGURES
Submitted by Rinaldo Ribeiro - http://www.sans.org/y2k/practical/Rinaldo_Ribeiro.doc

Detect Information: This trace was captured on the firewall.

Apr 10 02:07:51 kundun ipmon[1594]: 02:07:50.594491 ep1 @0:80 b 209.67.42.160,2900 -> x.y.z..33,53 PR tcp len 20 104 -S IN
Apr 10 02:07:51 kundun ipmon[1594]: 02:07:50.595656 ep1 @0:80 b 209.67.42.160,2901 -> x.y.z..33,53 PR tcp len 20 104 -S IN
Apr 10 02:07:51 kundun ipmon[1594]: 02:07:50.595771 ep1 @0:80 b 209.67.42.160,2902 -> x.y.z..33,53 PR tcp len 20 104 -S IN

Apr 10 03:02:28 kundun ipmon[1594]: 03:02:28.320739 ep1 @0:80 b 209.67.42.160,3800 -> x.y.z..12,53 PR tcp len 20 104 -S IN
Apr 10 03:02:28 kundun ipmon[1594]: 03:02:28.320848 ep1 @0:80 b 209.67.42.160,3801 -> x.y.z..12,53 PR tcp len 20 104 -S IN
Apr 10 03:02:28 kundun ipmon[1594]: 03:02:28.320952 ep1 @0:80 b 209.67.42.160,3802 -> x.y.z..12,53 PR tcp len 20 104 -S IN

Apr 10 03:13:13 kundun ipmon[1594]: 03:13:13.266251 ep1 @0:80 b 209.67.42.148,3000 -> x.y.z..12,53 PR tcp len 20 104 -S IN
Apr 10 03:13:13 kundun ipmon[1594]: 03:13:13.266858 ep1 @0:80 b 209.67.42.148,3001 -> x.y.z..12,53 PR tcp len 20 104 -S IN
Apr 10 03:13:13 kundun ipmon[1594]: 03:13:13.267027 ep1 @0:80 b 209.67.42.148,3002 -> x.y.z..12,53 PR tcp len 20 104 -S IN

Apr 10 05:24:49 kundun ipmon[1594]: 05:24:48.811312 ep1 @0:80 b 209.67.42.185,2900 -> x.y.z..12,53 PR tcp len 20 104 -S IN
Apr 10 05:24:49 kundun ipmon[1594]: 05:24:48.812251 ep1 @0:80 b 209.67.42.185,2901 -> x.y.z..12,53 PR tcp len 20 104 -S IN
Apr 10 05:24:49 kundun ipmon[1594]: 05:24:48.812369 ep1 @0:80 b 209.67.42.185,2902 -> x.y.z..12,53 PR tcp len 20 104 -S IN

Existence: Packets from net 209.67.42 are being sent to our servers' TCP 53 port.

History: Our firewall has been blocking packets sent from this network to port TCP 53 since the end of last year. It has been identified also packets sent to high udp ports, seeming to be originated by a "unix like" traceroute program.

Intent: The intent is collect RTT – "round trip time" information about how long it takes traffic to move between "client" and "server".

Technique: Three SYN packets are sent always in less than one second by a automated tool to TCP 53 port. The source port always increments by one and it starts between 2000 and 4000.

Analysis: This traffic is generated by a "global load-balancing system" called 3DNS, manufactured by F5 labs. In this case, everytime ours servers tried to resolve www.starmedia.com, some systems determine what is the closest server to my site sending packets as seen in this detect. It was considered by me, some time ago, as a "zone transfer" attempt and it could be done by another analyst if we had only a single packet. Some research was done to figure out all the picture:

1) who is 209.67.42.160?

   By accessing www.registro.br, maintained by FAPESP, the organization responsible for the domains in Brazil, we could find that the network 209.67.42 was registered in some starmedia.com's name servers.

2) getting in touch!

   The adminstrator of this domain was notified by e-mail and few days later I received a huge and complete answer about the "probe". It said "the traffic you're seeing can safely be ignored" and "if it's a real problem for you, I can hard-code your netblocks to our Brazil datacenter"... "that'll stop the probes".

3) looking at our logs again!

After this was much easier to find another "probe", done by another "RTT collect" tool and usually trying to access our server by a "unix like" traceroute before sending "TCP 53" packets.

Severity:

Criticality	5	Web servers
Lethality	1	Just getting the RTT.
System Countermeasures	5	All operating systems are running the latest patches and don't allow "zone transfers" from a name server that isn't a secondary.
Network Countermeasures	4	Firewall blocks this traffic.
Severity Score	-3	Severity = (Criticality + Lethality) – (system countermeasures + net countermeasures

SNMP is a very common scan on the Internet. This analyst took the time to analyze the targets and determined that they were primarily servers indicating this is a refinement in the prober's data about the site. We try to teach students to conduct Battle Damage Assessment (BDA) after successful reconnaissance probes.

RECON SNMP SCAN
Submitted by Ingeborg Ostrem Hellemo - http://www.sans.org/y2k/practical/Ingeborg_Hellemo.doc

udp: X.Y.Z.186 (5) (17)/161 23

Apr 10 09:47:46 gw.My.Net 60416: Apr 10 09:47:45.504: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.e.158(161), 1 packet
Apr 10 09:47:47 gw.My.Net 60417: Apr 10 09:47:46.984: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.c.188(161), 1 packet
Apr 10 09:47:48 gw.My.Net 60418: Apr 10 09:47:47.984: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.h.71(161), 1 packet
Apr 10 09:47:49 gw.My.Net 60419: Apr 10 09:47:48.984: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.g.59(161), 1 packet
Apr 10 09:47:53 gw.My.Net 60422: Apr 10 09:47:52.484: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.c.236(161), 1 packet
Apr 10 09:47:54 gw.My.Net 60423: Apr 10 09:47:53.496: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.g.80(161), 1 packet
Apr 10 09:47:56 gw.My.Net 60425: Apr 10 09:47:55.616: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4567) (ATM0/1/0.1 VC 26) -> a.b.c.246(161), 1 packet
Apr 10 09:47:57 gw.My.Net 60426: Apr 10 09:47:56.632: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4564) (ATM0/1/0.1 VC 26) -> a.b.c.236(161), 1 packet
Apr 10 09:47:59 gw.My.Net 60427: Apr 10 09:47:58.632: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.d.119(161), 1 packet
Apr 10 09:48:01 gw.My.Net 60428: Apr 10 09:48:00.180: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.d.101(161), 1 packet
Apr 10 09:48:03 gw.My.Net 60430: Apr 10 09:48:02.632: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4567) (ATM0/1/0.1 VC 26) -> a.b.h.71(161), 1 packet
Apr 10 09:48:05 gw.My.Net 60431: Apr 10 09:48:04.616: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4564) (ATM0/1/0.1 VC 26) -> a.b.g.80(161), 1 packet
Apr 10 09:48:27 gw.My.Net 60432: Apr 10 09:48:26.640: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.j.254(161), 1 packet
Apr 10 09:48:42 gw.My.Net 60433: Apr 10 09:48:41.452: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4563) (ATM0/1/0.1 VC 26) -> a.b.i.34(161), 1 packet
Apr 10 09:49:27 gw.My.Net 60434: Apr 10 09:49:26.108: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4566) (ATM0/1/0.1 VC 26) -> a.b.g.59(161), 1 packet
Apr 10 09:49:42 gw.My.Net 60435: Apr 10 09:49:41.364: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4565) (ATM0/1/0.1 VC 26) -> a.b.g.53(161), 1 packet
Apr 10 09:49:58 gw.My.Net 60436: Apr 10 09:49:57.200: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4567) (ATM0/1/0.1 VC 26) -> a.b.g.50(161), 1 packet
Apr 10 09:51:24 gw.My.Net 60437: Apr 10 09:51:23.673: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4564) (ATM0/1/0.1 VC 26) -> a.b.f.115(161), 1 packet
Apr 10 09:52:54 gw.My.Net 60438: Apr 10 09:52:53.669: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4566) (ATM0/1/0.1 VC 26) -> a.b.f.60(161), 1 packet
Apr 10 09:53:02 gw.My.Net 60439: Apr 10 09:53:01.501: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4565) (ATM0/1/0.1 VC 26) -> a.b.d.124(161), 1 packet
Apr 10 09:53:12 gw.My.Net 60440: Apr 10 09:53:11.345: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4567) (ATM0/1/0.1 VC 26) -> a.b.d.119(161), 1 packet
Apr 10 09:53:40 gw.My.Net 60441: Apr 10 09:53:39.301: %SEC-6-IPACCESSLOGP: list 111 denied udp X.Y.Z.186(4566) (ATM0/1/0.1 VC 26) -> a.b.d.101(161), 1 packet

Trace Information: This trace is from the log from our border Cisco router. First part of the trace is output from a script that searches for denied packets in the log. (Interpretation: 23 udp packets from 5 source ports on source address X.Y.Z.186 to port 161 on 17 different addresses) The second part is the relevant parts of the Cisco log itself. The border router denies incoming SNMP packets. Source address is .hscis.net. Target hosts are PCs and department servers.
Active Targeting: Yes. They try more than once, even though we do not allow inbound SNMP traffic.
History: None
Technique: Automated scan from 5 different source ports on remote host to selected hosts on 9 different subnets. Time span is 6 minutes, that is not very quick. The router does only deny packets to some ports, so other ports on the targets are possibly scanned at the same time. (I have not access to syslogs on these hosts to verify this.) The fact that mostly servers are scanned indicate that the attacker has collected information about our network earlier.
Intent: This is probably information gathering via SNMP, possibly for later use.
Severity: Low (0).

Criticality: 4 (servers and PCs)
Lethality: 2 (depends a bit on what they intend to use the information for)
System Countermeasures: 3 (unknown)
Net Countermeasures: 3 (router denies inbound SNMP)

This is one of the classics! Experts argue what exactly comprises the attack called winnuke, but it certainly includes the urgent flag set. This is an older attack, but still effective against older windows systems that do not have service pack upgrades.

WINNUKE
Submitted by Jason Steckler - http://www.sans.org/y2k/practical/Jason_Steckler.doc

14:32:43.464892 192.1.1.10.1269 > HOST.OUR.NET.139: S 109664:109664(0) win 8192 <mss 1460> (DF) [tos 0x10]
14:32:43.465149 HOST.OUR.NET.139 > 192.1.1.10.1269: S 39682:39682(0) ack 109665 win 8760 <mss 1460> (DF)
14:32:43.465343 192.1.1.10.1269 > HOST.OUR.NET.139: . ack 1 win 8760 (DF) [tos 0x10]
14:32:43.466977 192.1.1.10.1269 > HOST.OUR.NET.139: P 1:5(4) ack 1 win 8760 urg 4 (DF) [tos 0x10]
14:32:43.467201 HOST.OUR.NET.139 > 192.1.1.10.1269: FP 1:6(5) ack 5 win 8757 (DF)
14:32:43.467392 192.1.1.10.1269 > HOST.OUR.NET.139: . ack 7 win 8755 (DF) [tos 0x10]
14:32:43.467961 192.1.1.10.1269 > HOST.OUR.NET.139: R 109669:109669(0) win 0 (DF) [tos 0x10]
14:32:51.656210 192.1.1.10.1270 > HOST.OUR.NET.139: S 109670:109670(0) win 8192 <mss 1460> (DF) [tos 0x10]

Targeting: Yes
History: No previous history of this host on our net.
Techniques: This is a Winnuke attack against a Windows machine (port 139).
Intent: This is not a scan. This is an attempted DOS attack.
Analysis: This type of DOS attack is also called an Out of Band (OOB) nuke. The NetBIOS port (139) is being used on the target to send OOB traffic. If the target does not have the latest patches, lock-ups or Blue Screens can occur. The first 3 lines show the three-way handshake. Then traffic is sent with the Urgent bit set. In this case the client must be patched since the connection is discontinued. The attacker comes right back with his next port of 1270 and tries the same port on the target again. This probably indicates someone with a low level of knowledge or needs some more confirmation his tool isn't working. Another interesting point is that port 139 traffic from outside nets are not very common. On this particular net there should be no NetBIOS traffic coming from the outside. I can probably understand this type of traffic for a Windows NT trust or something that is established across two networks.

Components	Score	Comments
Criticality	2	Non-critical windows machine.
Lethality	4	If successful, a complete DOS.
System Countermeasures	5	This particular host is up to date with patches.
Network Countermeasures	2	Firewall present, but allowing port 139 traffic at the time of the incident.
Severity Total	-1	(Criticality + Lethality) – (System Countermeasures + Net countermeasures)

We want to be clear that Zone transfers are TCP port 53. That said, this trace illustrates that DMZ systems do get scanned quite consistently and need to be totally locked down. If possible, these systems should be on a screened subnet to reduce the number of probes that they must endure.

TARGETING DMZ
Submitted by Walter Grech - http://www.sans.org/y2k/practical/Walter_Grech.doc

	11-Apr-00	18:32:35 qfe0	33846	External.IP	D.M.Z.32	domain-udp
11-Apr-00	11-Apr-00	18:32:41 qfe0	33849	External.IP	D.M.Z.33	domain-udp
11-Apr-00	11-Apr-00	18:32:45 qfe0	33850	External.IP	D.M.Z.34	domain-udp
11-Apr-00	11-Apr-00	18:32:50 qfe0	33853	External.IP	D.M.Z.35	domain-udp
11-Apr-00	11-Apr-00	18:32:51 qfe0	33855	External.IP	D.M.Z.36	domain-udp
11-Apr-00	11-Apr-00	18:32:55 qfe0	33856	External.IP	D.M.Z.37	domain-udp
11-Apr-00	11-Apr-00	18:33:00 qfe0	33858	External.IP	D.M.Z.38	domain-udp
11-Apr-00	11-Apr-00	18:33:00 qfe0	33860	External.IP	D.M.Z.39	domain-udp
11-Apr-00						

Description:
This trace was targeting the DMZ side of a Firewall, the External.IP is scanning an address range for DNS services
Active Targeting:
Yes
History/Background/Methods:
This automated trace (timing is close) originating from the External.IP (not a busy machine, ports are almost sequential) is scanning our DMZ for a DNS server to respond. A Unix server doesn't necessarily have to have the prime role of a DNS server but just have the named deamon running with access allowed to port 53 (domain-udp). Zone transfers will allow a hacker to download specific host information about your systems. DNS servers are one of the primary "investigated" services by hackers.
Threat: Medium
Severity: Moderate
Subsequent Action:
Scanned the DMZ logs for past traffic from External.IP, none located. Checked existing DNS server for proper operation (All OK). Verify all other machines on the DMZ are NOT running named/DNS services. Verified the Firewall rulebase to allow domain udp/tcp traffic only to the public DNS server, drop all others (and log).

This trace and analysis is from the very educational submission by A. Jameson West, GCIA. In this case we see what life in cablemodem land is really like. Never connect to a cablemodem unless you have a personal firewall on the system and keep your patches up to date. Your box is certain to be tested.

RPC PORT PROBE
Submitted by A. Jameson West - http://www.sans.org/y2k/practical/A_Jameson_West.doc
source: BlackIce logs

RPC Port probe: This is a "Black Hat" attempt to access Sun Remote Procedure Call (RPC) (rpcbind, portmapper) services on your system. It is probably caused by a sweep of millions of machines on the Internet that are Sun Microsystems product running Solaris or SunOS, or some other compatible Unix OSs that support RPC service. If you are not one of these, and do not run an implementation of this on your Win.x or WinNT host, then you are not at risk. RPC (Remote Procedure Call) is a networking technology developed by Sun Microsystems. It is used on most UNIX machines, and is a prevalent way to build networked applications that use remote tasks for distributed processing, a forerunner of the client-server model architecture.

Its continued popularity persists due to legacy enterprise systems that still run these services and are still vulnerable to the same security pitfalls they have always endured. When a hacker probes for RPC services, it is an early indication of a hostile intent. If an RPC service is found on your system, then the next hacker step that should be expected is an RPC portmapper dump. This will list all RPC programs on your machine and enable the hacker to determine the vulnerabilities of each. After checking hacker sites for published lists of known exploits, the hacker can then be expected to attempt to break into your system using one of the exploits found.

BID detected 20 RPC port probes, of which 11 of originated from outside the *Cox@Home* domain. The 9 source IPs seen starting within inside Cox were considered benign as there was nothing extraordinary about them. All other remaining source IPs from outside Cox were unexpected and in some case, very stealthy. These were all considered hostile, and included the following:

- 202.30.26.72 with the BID backtrace name "mach.ajou.ac.kr";
- 216.227.17.17 with the BID backtrace name "dsl-216-227-17-12.chi.interchangedsl.com";
- 208.49.251.3 with no backtrace name from BID;
- 131.104.204.156 with no backtrace name from BID;
- 210.96.22.193 with no backtrace name from BID;
- 203.230.240.86 with no backtrace name from BID;
- 210.109.56.32 with no backtrace name from BID;
- 129.219.245.39 with the BID backtrace name "noname-13633.generic.asu.edu";
- 63.248.50.194 with the BID backtrace name "63-248-50-194.usa3.flashcom.net";
- 203.127.42.155 with no backtrace name from BID;
- 208.232.120.196 with no backtrace name from BID.

The brace of 7 unidentifiable RPC port probe requests with no traceable name reflects hostile RPC probing on the Cox network, and this remains unexplained.

Analyst David Hesprich, GCIA shares a bit about life in the @home network. If you are on this network, your systems are almost certainly being scanned for active NNTP and I believe SMTP ports as well. Note: the original trace has a small typo, it calls this traffic port 116, but we have corrected this for this publication.

ACCESS ATTEMPS TO PORT 119 (NNTP)
Submitted by David Hesprich - http://www.sans.org/y2k/practical/David_Hesprich.doc

Incident Summary

Multiple access attempts to port 119 (NNTP).

From "Global Incident Analysis Center: Detects Analyzed 3/26/00".

Log Data

```
Mar 22 13:23:36 box.at.victim.com /ipmon[4872]: 13:23:36.638478 le0 @0:19 b
24.0.94.130,38945 -> IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S Mar 22 13:23:37
box.at.victim.com /ipmon[4872]: 13:23:37.315117 le0 @0:19 b 24.0.94.130,38945 ->
IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R Mar 22 13:23:37 box.at.victim.com /ipmon[4872]:
13:23:37.326922 le0 @0:19 b 24.0.94.130,39317 -> IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S
Mar 22 13:23:38 box.at.victim.com /ipmon[4872]: 13:23:38.312697 le0 @0:19 b
24.0.94.130,39317 -> IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R Mar 22 18:04:02
box.at.victim.com /ipmon[4872]: 18:04:01.661131 le0 @0:19 b 24.0.94.130,59273 ->
IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S Mar 22 18:04:03 box.at.victim.com /ipmon[4872]:
18:04:03.188819 le0 @0:19 b 24.0.94.130,59273 -> IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R
Mar 22 18:04:03 box.at.victim.com /ipmon[4872]: 18:04:03.210320 le0 @0:19 b
24.0.94.130,60187 -> IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S Mar 22 18:04:04
box.at.victim.com /ipmon[4872]: 18:04:03.811455 le0 @0:19 b 24.0.94.130,60187 ->
IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R Mar 22 22:48:17 box.at.victim.com /ipmon[4872]:
22:48:16.835881 le0 @0:19 b 24.0.94.130,50230 -> IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S
Mar 22 22:48:18 box.at.victim.com /ipmon[4872]: 22:48:18.161571 le0 @0:19 b
24.0.94.130,50230 -> IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R Mar 22 22:48:18
box.at.victim.com /ipmon[4872]: 22:48:18.173064 le0 @0:19 b 24.0.94.130,50678 ->
IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S Mar 22 22:48:19 box.at.victim.com /ipmon[4872]:
22:48:18.986828 le0 @0:19 b 24.0.94.130,50678 -> IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R
Mar 23 03:20:08 box.at.victim.com /ipmon[4872]: 03:20:07.585219 le0 @0:19 b
24.0.94.130,62610 -> IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S Mar 23 03:20:08
box.at.victim.com /ipmon[4872]: 03:20:08.045238 le0 @0:19 b 24.0.94.130,62610 ->
IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R Mar 23 03:20:08 box.at.victim.com /ipmon[4872]:
03:20:08.056194 le0 @0:19 b 24.0.94.130,62931 -> IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S
Mar 23 03:20:09 box.at.victim.com /ipmon[4872]: 03:20:08.858156 le0 @0:19 b
24.0.94.130,62931 -> IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R Mar 23 07:46:43
box.at.victim.com /ipmon[4872]: 07:46:42.379020 le0 @0:19 b 24.0.94.130,61302 ->
IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S Mar 23 07:46:44 box.at.victim.com /ipmon[4872]:
07:46:43.418154 le0 @0:19 b 24.0.94.130,61302 -> IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R
Mar 23 07:46:44 box.at.victim.com /ipmon[4872]: 07:46:43.442809 le0 @0:19 b
24.0.94.130,62218 -> IP.OF.VICTIM.COM,119 PR tcp len 20 44 -S Mar 23 07:46:44
box.at.victim.com /ipmon[4872]: 07:46:44.091157 le0 @0:19 b 24.0.94.130,62218 ->
IP.OF.VICTIM.COM,119 PR tcp len 20 40 -R
```

Protagonist

Obfuscated @Home user	IP.OF.VICTIM.COM

Antagonist

authorized-scan.security.home.net	24.0.94.130

Analysis

This trace almost certainly sourced from an @Home user.

@Home is an Internet service that provides high bandwidth Internet access over cable TV cables.

@Home has a good abuse policy but often does not enforce it. This allows spammers based at @Home to run rampant. In addition, computers connected to the Internet via @Home are often misconfigured to act as relays. These are quickly discovered by spammers and used to relay spam. As part of the measures to avoid a Usenet Death Penalty (UDP), @Home is scanning its users for open NNTP port relays.

Similar scans have been submitted to GIAC: http://www.sans.org/y2k/030500.htm on March 5, 2000; http://www.sans.org/y2k/030800.htm on March 8, 2000; http://www.sans.org/y2k/031000.htm on March 10, 2000; and http://www.sans.org/y2k/032800.htm on March 28, 2000 900.

This next trace is an excellent example of a common pattern many analysts will run into. ICQ is a very popular software product.

ICQ
Submitted by Holly Shepardson - http://www.sans.org/y2k/practical/Holly_Shepardson.doc

```
04/01/2000 16:23:43.768  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 16898, LAN - "  -  Rule
0
04/01/2000 16:24:23.864  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 16898, LAN - "  -  Rule
0
04/01/2000 16:26:44.496  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 14272, LAN - "  -  Rule
0
04/01/2000 16:27:24.608  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 14272, LAN - "  -  Rule
0
04/01/2000 16:30:05.384  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 12082, LAN - "  -  Rule
0
04/01/2000 16:30:25.416  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 12082, LAN - "  -  Rule
0
04/01/2000 16:32:47.912  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 21304, LAN - "  -  Rule
0
04/01/2000 16:33:07.928  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 21304, LAN - "  -  Rule
0
04/01/2000 16:33:18.016  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 21304, LAN - "  -  Rule
0
04/01/2000 16:35:37.032  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 18322, LAN - "  -  Rule
0
04/01/2000 16:35:55.048  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 18322, LAN - "  -  Rule
0
04/01/2000 16:36:07.064  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 18322, LAN - "  -  Rule
0
04/01/2000 16:38:48.496  UDP packet dropped -  "Source:205.188.153.101, 4000, WAN - "  "Destination:user.mynet, 16668, LAN - "  -  Rule
0
```

Active Targeting:
Yes.

History:
I have seen this subnet before with the same source port. The subnet belongs to mirabilis.com and port 4000 is used for ICQ. In order for ICQ to work through firewalls port 4000 must be opened for bi-directional communication to icq.mirabilis.com. Our firewall is blocking the communication coming from the ICQ server.

Intent:
The ICQ server is trying to establish a connection with one of my users who is using an ICQ client. This detect has a low severity level as far as someone attempting unauthorized access to our network. However, once again this is an unapproved Internet activity for some of our users and more monitoring is required for our internal network.

If there was a top ten vulnerabilities list (and there soon will be) CGI-BIN would have to be on it. There is no end to the number and variety of attacks that target programs on web servers. Even the oldest (phf, php) continue to show up with new variations.

CGI VULNERABILITY
Submitted by Ron Bartninkas - http://www.sans.org/y2k/practical/Ron_Bartninkas.doc

I noticed this in the log file for the web server running on port 80 on my proxy server. The purpose of this port 80 web server is to serve up web proxy auto configuration files. The message in the errors file indicates that this GET failed. The proxy server is at my.net.128.224.

Web server logs:
access:205.215.135.210 - - [09/Apr/2000:00:52:17 -0400] "GET /cgi-bin/view-source?cgi-bin/view-source HTTP/1.0" 404 207

errors:[09/Apr/2000:00:52:18] warning: for host 205.215.135.210 trying to GET /cgi-bin/view-source, send-file reports: can't find /usr/local/netscape/suitespot/docs/cgi-bin/view-source (No such file or directory)

Active Targeting: Yes

Intent: Searching for cgi-bin/view-source vulnerability (see www.cz.nessus.org/plugins/CGI_abuses/view_source.html). A look at a second web server shows the same get and failure

Technique: At first it appears targeted at the web proxy server. Further investigation reveals that this was a targeted sweep of known web servers (see netflow data below). This was targeted at the specific hosts and must have been driven from a previous recon that identified the targeted web servers. The targeting is evidenced by the incrementing port (dstport in the netflow data) of the sweeper (205.215.135.210). The "firstflowstamps" indicate many packets sent in a few seconds – some sort of scanning program. All the connections are to port 80 and all are TCP.

There were some hosts that were not in the list (my.net.8.11, 8.12, 8.13) the prober had, and some that were non-responsive as seen by the missing dstports after 60187.

History: Looking at the history for the day in question (April 9, 2000), more activity from 205.215.135.210 was seen later in the day. There were 58 connections to the my.net.128.224 between 17:29:24 and 17:31:32 from 205.215.135.210 all to port 56735(UDP). There was no prior history relating to IP 205.215.135.210 until this day.

Existence: 205.215.135.210 belongs to Minneapolis Public Schools. Either this machine has been compromised or there was a group of students playing. This latter case is highly unlikely as the incident started around 00:51:54 on a Sunday morning. I doubt the schools were open. The machine at IP 205.215.135.210 appears to be the DNS server and mail server for the schools (based on whois info for NET 205.215.135 and MX records for the domain mpls.k12.mn.us). This has been reported to the coordinator for 205.215.135.x. A phone call to the coordinator will be made if no response to the email is received within a week.

Severity:
Criticality – 4 – the my.net.8.x addresses are switches with web administration capability

Lethality – 2 – the GET failed for proxy and the above mentioned switches prompt for a password
System Countermeasures – 3 – switches have 2 levels of passwords
Network Countermeasures – 1 – no firewall
(4+2)-(3+1)=2

Analysis: This was targeted at a list of web servers trying to identify those which may have the "cgi-bin/view-source" vulnerability.

Netflow Data:
(srcip|dstip|srcport|dstport|protocol|tos|packets|bytes|flows|firstflowstamp|lastflowstamp|totalactivetime-ms)

my.net.3.229|205.215.135.210|80|**60036**|6|0|5|808|2|955255914|955255925|2364
my.net.3.231|205.215.135.210|80|**60037**|6|0|4|275|3|955255915|955255943|0
my.net.8.6|205.215.135.210|80|**60038**|6|0|6|249|2|955255914|955255925|4352
my.net.8.7|205.215.135.210|80|**60039**|6|0|5|894|3|955255914|955255931|1152
my.net.8.8|205.215.135.210|80|**60040**|6|0|4|165|2|955255914|955255925|12
my.net.8.9|205.215.135.210|80|**60041**|6|0|6|249|4|955255915|955255937|12
my.net.8.10|205.215.135.210|80|**60042**|6|0|7|290|3|955255914|955255929|4344
my.net.8.14|205.215.135.210|80|**60043**|6|0|5|205|2|955255915|955255925|2344
…
my.net.8.196|205.215.135.210|80|**60186**|6|0|4|509|3|955255915|955255922|8
my.net.8.197|205.215.135.210|80|**60187**|6|0|7|1323|5|955255915|955255953|2072
my.net.21.7|205.215.135.210|80|**60192**|6|0|5|654|1|955255915|955255917|2584
my.net.21.9|205.215.135.210|80|**60193**|6|0|5|455|1|955255915|955255917|2572
…
my.net.33.25|205.215.135.210|80|**60201**|6|0|5|902|2|955255915|955255917|1440
my.net.35.68|205.215.135.210|80|**60202**|6|0|4|768|1|955255915|955255917|2576
…
my.net.111.203|205.215.135.210|80|**60291**|6|0|6|614|3|955255915|955255923|1248
my.net.**128.224**|205.215.135.210|80|**60296**|6|0|4|329|3|955255915|955255939|20
my.net.139.23|205.215.135.210|80|**60302**|6|0|6|976|2|955255915|955255923|3860

2'nd Web Server Log:
access_log:ultra.mpls.k12.mn.us - - [09/Apr/2000:00:51:54 -0400] "GET /cgi-bin/view-source?cgi-bin/view-source HTTP/1.0" 404 172
error_log:[Sun Apr 9 00:51:54 2000] access to /usr/local/etc/httpd/cgi-bin/view-source failed for ultra.mpls.k12.mn.us, reason: script not found or unable to stat

One of the great things about intrusion detection is you can't go a week without seeing a new pattern. The trace and analysis shown will get you up to speed on WebAmp.

WEBAMP
Submitted by Jerry Litteer - http://www.sans.org/y2k/practical/GeraldLitteer.html

Subject: **WebAmp** traffic

SRC	weavnt (Windows-NT)
DST	g2lb.spinner.com, spinlb1.spinner.com service, 205.188.246.121, g2mh32-qfe0.spinner.com

A few days after the latest WEB radio client **WebAmp** was announced, we saw what appeared to be massive udp scans of our internal class B network. Several rejected udp records were found in our firewall log. At first glance this appeared to be a udp scan of serveral hosts within our internal class B network.
What is really happening is:

1. Client requests music radio via http request to central server.
2. Server responds to client via exchange from tcp-554 to tcp-903 with url location of music channel
3. Client issues tcp-554 request to radio server.
4. Radio server responds witn udp-6970 packet containing audio.

Note that the location of the final server is determined by a load sharing algorithm and can change with each request. If the udp packets are blocked at the firewall this appears to be a coordinated udp host scan.
The first log is from the firewall:

5:25:31	accept	fw>out	proto	tcp	src	weavnt	dst	spinlb1.spinner.com	service	http	s_port	1900 len 44 rule 25
5:25:32	accept	fw>out	proto	tcp	src	weavnt	dst	205.188.246.121	service	554	s_port	1903 len 44 rule 25
5:25:34	reject	fw>in	proto	udp	src	g2mh32-qfe0.spinner.com	dst	weavnt	service	6970	s_port	29881 len 647 rule 28

This next dumps are the snoop output of the tcp-554 backchannel packet (not seen in the firewall logs) and the rejected udp-6970 packet.

```
Frame (37 on wire, 317 captured)
Frame arrived on Jan 25, 2000 5:25:01.9423
Total frame length: 37 bytes
Capture frame length: 37 bytes
Ethernet II
Destination: 08:00:20:9d:cd:fd (Sun_9d:cd:fd)
Source: 00:e0:34:54:9c:00 (Cisco_54:9c:00)
Type: IP (0x0800)
Internet Protocol
Version: 4
Header length: 20 bytes
Type of service: 0x00 (None)
000. .... = routine precedence
...0 .... = normal delay
.... 0... = normal throughput
.... .0.. = normal reliability
.... ..0. = normal cost
Total length: 303
Identification: 0xafe
Flags: 0x4
... .... = don't fragment
..0. .... = last fragment
Fragment offset: 0
```

```
Time to live: 245
Protocol: TCP
Header checksum: 0xe42
Source address: 205.88.246.121 (205.188.246.121)
Destination address: weavnt (x.y.59.94)
Transmission Control Protocol
Source port: 554 (554)
Destination port: 903 (1903)
Sequence number: 242365995
Acknowledgement number: 4075
Header length: 20 bytes
Flags: 0x8
..0. .... = No urgent pointer
... .... = Acknowledgment
.... ... = Push
.... .0.. = No reset
.... ..0. = No Syn
.... ...0 = No Fin
Window size: 8760
Checksum: 0x348a
Data (263 bytes)

   0  5254 5350 2f3 2e30 2032 3030 204f 4b0d   RTSP/1.0 200 OK.
  0  0a43 5365 713a 2035 0d0a 4461 7465 3a20   .CSeq: 5..Date:
  20  5475 652c 2032 3520 4a6 6e20 3230 3030   Tue, 25 Jan 2000
  30  2032 323a 3235 3a33 3420 474d 540d 0a52    22:25:34 GMT..R
  40  5450 2d49 6e66 6f3a 2075 726c 3d72 7473   TP-Info: url=rts
  50  703a 2f2f 6732 7272 2e73 7069 6e6e 6572   p://g2rr.spinner
  60  2e63 6f6d 3a35 3534 2f73 7069 6e6c 696e   .com:554/spinlin
  70  6b73 2f33 3834 2d4c 6974 655f 526f 636b   ks/384-Lite_Rock
  80  5f47 322d 323 3234 3635 3637 3330 2e72    _G2-2124656730.r
  90  6d2f 7374 7265 66d 6964 3d30 3b73 6571    m/streamid=0;seq
  a0  3d30 3b72 7470 7469 6d65 3d30 0d0a 0d0a   =0;rtptime=0....
  b0  5254 5350 2f3 2e30 2034 3531 2050 6172   RTSP/1.0 451 Par
  c0  66d 6574 6572 204e 6f74 2055 6e64 6572   ameter Not Under
  d0  7374 6f6f 640d 0a43 5365 73a 2036 0d0a   stood..CSeq: 6..
  e0  446 7465 3a20 5475 652c 2032 3520 4a61   Date: Tue, 25 Ja
  f0  6e20 3230 3030 2032 323a 3235 3a33 3420   n 2000 22:25:34
 00  474d 540d 0a0d 0a                         GMT....


Frame (652 on wire, 652 captured)
Frame arrived on Jan 25, 2000 5:25:02.0742
Total frame length: 652 bytes
Capture frame length: 652 bytes
Ethernet II
Destination: 08:00:20:9d:cd:fd (Sun_9d:cd:fd)
Source: 00:e0:34:54:9c:00 (Cisco_54:9c:00)
Type: IP (0x0800)
Internet Protocol
Version: 4
Header length: 20 bytes
Type of service: 0x00 (None)
000. .... = routine precedence
...0 .... = normal delay
.... 0... = normal throughput
.... .0.. = normal reliability
```

```
.... ..0. = normal cost
Total length: 638
Identification: 0xaff
Flags: 0x4
... .... = don't fragment
..0. .... = last fragment
Fragment offset: 0
Time to live: 245
Protocol: UDP
Header checksum: 0xe2ec
Source address: g2mh32-qfe0.spinner.com (205.88.246.83)
Destination address: weavnt (x.y.59.94)
User Datagram Protocol
    Source port: 2988 (29881)
    Destination port: 6970 (6970)
    Length: 68
    Checksum: 0xcfaa
Data (60 bytes)

   0  4000 0247 0000 0d0 0000 2e9f 30af 45f7   @..G........0.E.
  0  1f94 97b6 eef0 8b28 2d61 0239 7ce5 c26d   ........(-a.9|..m
  .    .... .... .... .... .... .... .... ....
  .    .... .... .... .... .... .... .... ....
  .    .... .... .... .... .... .... .... ....
 250  c9ea fa35 00a 47c6 f78f ffc7 f18e 7115   ...5..G.......q.
 260  7d72                                     }r
```

Scanning or connections to and from vendors is not unheard of. Already in this document we have seen an example of a software product with hardcoded Internet Addresses for SNMP traps. RealAudio bothers my personal firewall every couple days because it is asking for permission to connect back home. Here is an example of a detect similar to the Pkware example we saw earlier where the vendor comes calling.

COMPAQ, hmm...
Submitted by Shane Boothe - http://www.sans.org/y2k/practical/Shane_Boothe.doc

Time	Source	Destination	Protocol	Info
23:03:19.9379	206.251.4.210	cablemodem.net	UDP	Source port: 1070 Destination
09:53:45.7975	206.251.4.210	cablemodem.net	UDP	Source port: 1045 Destination
09:53:51.5325	206.251.4.210	cablemodem.net	UDP	Source port: 1031 Destination
09:54:01.8435	206.251.4.210	cablemodem.net	UDP	Source port: 1077 Destination
09:54:14.5085	206.251.4.210	cablemodem.net	UDP	Source port: 1075 Destination
09:54:23.7125	206.251.4.210	cablemodem.net	UDP	Source port: 1072 Destination
01:26:09.8009	206.251.4.210	cablemodem.net	UDP	Source port: 1055 Destination
01:26:13.1410	206.251.4.210	cablemodem.net	UDP	Source port: 1050 Destination
01:26:19.0080	206.251.4.210	cablemodem.net	UDP	Source port: 1033 Destination
01:26:29.6579	206.251.4.210	cablemodem.net	UDP	Source port: 1074 Destination
01:26:39.9379	206.251.4.210	cablemodem.net	UDP	Source port: 1031 Destination
01:26:51.1130	206.251.4.210	cablemodem.net	UDP	Source port: 1053 Destination
03:52:48.6009	206.251.4.210	cablemodem.net	UDP	Source port: 1071 Destination
03:52:51.4320	206.251.4.210	cablemodem.net	UDP	Source port: 1067 Destination
03:52:57.4079	206.251.4.210	cablemodem.net	UDP	Source port: 1067 Destination
03:53:07.0520	206.251.4.210	cablemodem.net	UDP	Source port: 1075 Destination
03:53:18.1679	206.251.4.210	cablemodem.net	UDP	Source port: 1069 Destination
03:53:29.7580	206.251.4.210	cablemodem.net	UDP	Source port: 1077 Destination
06:18:04.8079	206.251.4.210	cablemodem.net	UDP	Source port: 1060 Destination
06:18:08.1380	206.251.4.210	cablemodem.net	UDP	Source port: 1059 Destination
06:18:12.2910	206.251.4.210	cablemodem.net	UDP	Source port: 1078 Destination
06:18:22.4620	206.251.4.210	cablemodem.net	UDP	Source port: 1031 Destination
06:18:37.6380	206.251.4.210	cablemodem.net	UDP	Source port: 1060 Destination
06:18:48.6879	206.251.4.210	cablemodem.net	UDP	Source port: 1067 Destination

History	Taken from a friend's Compaq Presario attached to a cable modem. The above scans were very common and time and frequency pattern.	
Active Targeting?	Yes.	
Criticality	5	Home computer with personal/financial data on it.
Lethality	3	Not sure how to rate this one, but since system updated may be installed by this, I gave it a 3.
System Countermeasures	4	OS is up to date.
Network Countermeasures	4	Host-based firewall installed.
Severity	0	Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
Notes	After seeing the above activity I did an nslookup on 206.251.4.210, which returned Compaq as the owner. Be would scan their customer's computers, I did a little research. UPD 371 is associated with either Clearcase, w from Rational Software (http://www.rational.com/products/clearcase/index.jtmpl) or Backweb (http://www.b based software distribution solution. A quick search of Backweb's site verified the Compaq relationship (http://www.backweb.com/html/compaq.html). On the client computer an application named *Compaq Servic* started upon boot up. The software allows Compaq to deliver software updates and patches automatically. I'r any case we configured the firewall to trust this address so the updates could be delivered.	

Odd size fragments to an obscure port, from multiple source addresses all with the same TTL, what else could you ask for in a detect! Note that a probe against discard if successful would indicate a poorly protected Unix system.

POSSIBLE PORT SCAN?
Submitted by Stephen Pohler - http://www.sans.org/y2k/practical/Stephen_Pohler.doc

date: 4-6-2000

time	src.ip.addr src.port		dst.ip.addr dst.port	type size	frag id size@offset	ttl
14:38:23.136210	10.1.2.252	>	10.9.8.38:		(frag 20273:481@552)	(ttl 111)
14:39:28.033346	10.1.2.252.4274	>	10.9.8.38.9:	udp 1025	(frag 35123:552@0+)	(ttl 111)
14:39:28.327999	10.1.2.252	>	10.9.8.38:		(frag 35123:481@552)	(ttl 111)
15:17:52.125341	10.3.4.239	>	10.9.8.38:		(frag 54539:481@552)	(ttl 111)
15:18:40.506071	10.3.4.239.1045	>	10.9.8.38.9:	udp 1025	(frag 51468:552@0+)	(ttl 111)
15:18:40.675600	10.3.4.239	>	10.9.8.38:		(frag 51468:481@552)	(ttl 111)
15:19:44.338447	10.3.4.239.1040	>	10.9.8.38.9:	udp 1025	(frag 11790:552@0+)	(ttl 111)
15:19:44.524989	10.3.4.239	>	10.9.8.38:		(frag 11790:481@552)	(ttl 111)

date: 4-7-2000

time	src.ip.addr src.port		dst.ip.addr dst.port	type size	frag id size@offset	ttl
15:19:39.722456	10.5.6.123	>	10.9.8.38:		(frag 41749:481@552)	(ttl 111)

date: 4-11-2000

time	src.ip.addr src.port		dst.ip.addr dst.port	type size	frag id size@offset	ttl
10:34:57.231266	10.7.8.130	>	10.9.8.38:		(frag 5279:481@552)	(ttl 109)
10:35:13.723237	10.7.8.130.1755	>	10.9.8.38.9:	udp 1025	(frag 24735:552@0+)	(ttl 109)
10:35:13.849714	10.7.8.130	>	10.9.8.38:		(frag 24735:481@552)	(ttl 109)
10:35:29.509087	10.7.8.130.1760	>	10.9.8.38.9:	udp 1025	(frag 50079:552@0+)	(ttl 109)
10:35:29.659348	10.7.8.130	>	10.9.8.38:		(frag 50079:481@552)	(ttl 109)
10:35:45.568734	10.7.8.130.1764	>	10.9.8.38.9:	udp 1025	(frag 12448:552@0+)	(ttl 109)
10:35:45.714657	10.7.8.130	>	10.9.8.38:		(frag 12448:481@552)	(ttl 109)
10:36:41.505696	10.11.12.90	>	10.9.8.38:		(frag 18022:481@552)	(ttl 110)
10:37:42.855458	10.13.14.157	>	10.9.8.38:		(frag 11531:481@552)	(ttl 13)
10:38:30.344990	10.13.14.157.2336	>	10.9.8.38.9:	udp 1025	(frag 11020:552@0+)	(ttl 13)
10:38:30.465881	10.13.14.157	>	10.9.8.38:		(frag 11020:481@552)	(ttl 13)
10:38:41.836356	10.15.16.76	>	10.9.8.38:		(frag 62730:481@552)	(ttl 114)
10:39:18.859179	10.13.14.157	>	10.9.8.38:		(frag 13069:481@552)	(ttl 13)
10:39:42.525005	10.13.14.157.2332	>	10.9.8.38.9:	udp 1025	(frag 46093:552@0+)	(ttl 13)
10:39:42.636211	10.13.14.157	>	10.9.8.38:		(frag 46093:481@552)	(ttl 13)
10:39:58.345322	10.13.14.157.2338	>	10.9.8.38.9:	udp 1025	(frag 1038:552@0+)	(ttl 13)
10:39:58.456898	10.13.14.157	>	10.9.8.38:		(frag 1038:481@552)	(ttl 13)
10:40:14.308545	10.13.14.157.2330	>	10.9.8.38.9:	udp 1025	(frag 20494:552@0+)	(ttl 13)
10:40:14.418730	10.13.14.157	>	10.9.8.38:		(frag 20494:481@552)	(ttl 13)

analysis:

- existence first instance of these IP addresses; only traffic are these fragments, and no subsequent traffic
  - WHOIS: These IP address spaces are owned by:

-UUNET Canada Inc.
-Cable & Wireless USA
-VESTELNET Elektronik]letisim ve Bilgilendirme A.S. in Istanbul, Turkey
-Free Net Corporation Pty. Ltd. In Wollongong, Australia
-U S WEST Communications Svcs, Inc. in Minneapolis, Minnesota
-Level 3 Communications, LLC in Louisville, CO

- history recent addition of a research network; logged using windump on NT
- targeting yes
- technique never a stimulus for these responses
  -10.9.8.38 is an IP address that was never allocated
  -Port 9 is the discard service, which is never enabled
  - similar fragment patterns:
  -'legal' fragments - in that they will reassemble; except the missing first fragments
    -fragment ids match and offsets align properly
    -a first fragment with the more fragments bit set and then a final fragment
    -according to TCP/IP Illustrated v1, all fragments except the final fragment must be on an 8-byte boundary (552 ÷ 8 = 69; so it is on an 8-byte boundary)
  - odd number of packets from each server:
  -first packet from each server is always the 481@552 offset of the fragmented packet
  - udp size of the packets are all the same:
  -also, in the first packet, the 552 bytes of IP data minus the 8-byte UDP header is 544 bytes of udp data, and the final fragment is 481 bytes of data…481 + 544 = 1025; which is the correct udp data size
  - probably non-spoofed source addresses:
  -ttls are widely variable, indicating geographically different locations
  -packet delivery times are variable, indicating different server loading and OSes
- intention unknown intention
  -hacker testing a mapping technique to elicit an icmp port unreachable message?
- outcome all packets were dropped at the firewall, as port 9 (discard) is not an allowed service
- comments Fragmented packets from around the world with similar fragment sizes and udp packet sizes targeting an IP address that was never allocated and a service which we do not allow
  Possibly the hacker intends for the first fragment to be accepted as a part of an ongoing, established connection, and then the following packets from the same source to the same IP address will be passed.

- severity (Criticality + Lethality) - (System + Network) = Severity

(2 + 1) - (5 + 5) = -7

This next trace is a great example of running a detect down and then by sharing it, Eric Gallagher, GCIA makes us all a little smarter.

PORT 1600 SCAN
Submitted by Eric Gallagher - http://www.sans.org/y2k/practical/Eric_Gallagher.doc

From local tcpdump:

21:57:18.468912 MY.NET.2.4.4277 > MY.NET.1.0.1600: udp 46
21:59:18.478004 MY.NET.2.4.4278 > MY.NET.1.0.1600: udp 46
22:01:18.487730 MY.NET.2.4.4279 > MY.NET.1.0.1600: udp 46
22:03:18.498327 MY.NET.2.4.4280 > MY.NET.1.0.1600: udp 46
22:05:18.507726 MY.NET.2.4.4281 > MY.NET.1.0.1600: udp 46
22:07:18.517192 MY.NET.2.4.4282 > MY.NET.1.0.1600: udp 46
22:09:18.527483 MY.NET.2.4.4283 > MY.NET.1.0.1600: udp 46
22:11:18.536690 MY.NET.2.4.4284 > MY.NET.1.0.1600: udp 46
22:13:18.546665 MY.NET.2.4.4285 > MY.NET.1.0.1600: udp 46

This scan repeated itself any number of times on my networks. The source machine ran through its ports in order, always broadcasting (in the older, BSD-style way, at the network address) to destination port 1600. Since I am fairly new to this particular job and therefore unfamiliar with these networks, I didn't recognize the source machine address offhand, although I discovered it was a legitimate node on my network. It is an older medical imaging device manufactured by General Electric. Recently, it has been moved to a new room and turned back on after a long period of downtime.

Targeting: This much is obvious, since the source machine is sending to the network address of the entire MY.NET.1 segment. The nature of the response expected is a mystery but the scan is targeted to my networks, albeit in a primitive way.

Intent: A scan for port 1600 looks suspicious for a couple of reasons. First, there is a trojan called Shivka-Burka that uses this UDP port. I'm seeing only the broadcast traffic on our network (because we're switched) but this could be one side of a two-way communication with a Shivka-Burka trojan. Of course, this is such a loud, obvious scan that the Shivka-Burka scenario seems unlikely but it can't be ruled out. The second point of suspicion is the ISSD service that's registered as the legitimate service user of port 1600. If MY.NET.2.4 is a compromised machine, the cracker could have set up a scan for this service in order to exploit a weakness in it.

Techniques: A typical broadcast mapping attempt, albeit on an unusual port.

History: The network mapping restarted every time this particular medical console was turned back on. Unfortunately, none of the GE technicians had any idea why the machine was doing this. Some of the more experienced medical/technical workers felt that we were seeing a hostile signature. Certainly the machine had been around long enough to get cracked (nine years).

When I came to this department, four Solaris machines had been lost to successful cracks and more were under attack. However, no other old SunOS system had not been cracked during the months of network break-ins and it seemed likely that a network hacker/cracker smart enough to get into one would have reached some of the others, as well. Since the MY.NET.2.4 machine was based on SunOS, it seemed, due to our local, historical reasons, more safe than a Solaris or IRIX based station.

I took a look, via nmap, at the interesting ports left open on MY.NET.2.4. Among them were:

Port	State	Service
104/tcp	open	acr-nema
655/tcp	open	unknown
658/tcp	open	unknown
670/tcp	open	unknown
723/tcp	open	unknown
725/tcp	open	unknown

727/tcp open unknown
728/tcp open unknown
733/tcp open unknown
736/tcp open unknown
1024/tcp open unknown

So the nmap tool did not detect an open port 1600 on the MY.NET.2.4 machine itself. This was not what I expected and increased my concern, since I felt that this medical console would be configured like all others from GE and would therefore have a service open on port 1600 if it was a legitimate destination port. The port 104 is a medical standard for DICOM data transfer, so that was expected. The unknown ports seem to be specific to this older generation of GE equipment, so they were not a serious cause for alarm. At this point, however, I had to step up my severity estimate.

Severity: Moderate. To understand why, you have to know the full history. Although I did not see any crack attempt (no lethality) and I felt the crude nature of the scan revealed its innocence, I had to increase my judgement of the scan's severity the longer this activity remained unsolved. GE's apparent ignorance of the network software along with the nature of the data (medical and therefore confidential) necessitated further investigation.

Action Recommendation: The first action would be to talk with the manufacturer's representatives and determine whether or not this is expected behavior. A check for rootkits (although often futile) must be done, as should a check for all other signs of intrusion into the GE medical workstation. On the IDS end, it's possible to check to see if the packets were corrected formed or showed any signs of tampering. It's an even better idea to monitor port 1600 activity on all machines in the network and turn off the GE medical workstation if there is any sign of attack through this port.

Resolution: I failed to find any clear signs of a break-in on this machine and had determined that the UDP packets looked legitimate based on this pattern:

```
16:25:08.759896 ic1.3701 > MY.NET.1.0.1600: udp 46
        4500 004a 60ee 0000 3c11 72e1 80e7 d504
        80e7 d400 0e75 0640 0036 0000 0000 007e
        0000 05e8 4e4d 5231 0000 4943 3000 0000
        0000 0000 0000
16:27:08.769361 ic1.3702 > MY.NET.1.0.1600: udp 46
        4500 004a 60ef 0000 3c11 72e0 80e7 d504
        80e7 d400 0e76 0640 0036 0000 0000 007e
        0000 05e8 4e4d 5231 0000 4943 3000 0000
        0000 0000 0000
```

Not only were the packets correctly formed but, as far as UDP was concerned, they were all identical. Above, you can see that three octets increment or decrement regularly. These are all part of the IP datagram. The change from 60ee to 60ef represents the changing ID number of the datagram. The decrement from 72e1 to 72e0 represents the differences in IP header checksum. The change from 0e75 to 0e76 shows the IP destination address incrementing, which we can already see in our earlier trace. I admit I don't understand the IP header checksum business but the behavior seems consistent. All further packets display the same properties:

```
16:31:08.790067 ic1.3704 > MY.NET.1.0.1600: udp 46
        4500 004a 60f1 0000 3c11 72de 80e7 d504
        80e7 d400 0e78 0640 0036 0000 0000 007e
        0000 05e8 4e4d 5231 0000 4943 3000 0000
        0000 0000 0000
16:33:08.799578 ic1.3705 > MY.NET.1.0.1600: udp 46
        4500 004a 60f2 0000 3c11 72dd 80e7 d504
        80e7 d400 0e79 0640 0036 0000 0000 007e
        0000 05e8 4e4d 5231 0000 4943 3000 0000
        0000 0000 0000
```

In the end, my judgement was that this traffic consisted of legitimate broadcast packets. I could find no signs of a break-in on the medical workstation, although I found myself somewhat unfamiliar with what it means to have a normal machine environment in an old GE medical system.

When I mentioned that the GE medical workstation was running NIS services (which looked more strange than suspicious, to me), this helped one of the GE technicians. In fact, the very first GE technician I'd asked about the port scan came back to me after doing a bit of research and some plain-old remembering. He said that port 1600 was used by the old GE AdvantageNet for communication between medical imaging stations as part of the coordination of traffic flow between them. An old Genesis master console not only scans port 1600 for other GE AdvantageNet devices but administer them as an NIS server as well, although data transfers via the DICOM standard take place over port 104, not over any of the GE or NIS ports. Since the MY.NET.2.4 machine had been a master console when it was originally active, it ran the scans and the NIS services.

I still do not know the purpose of all the other GE-specific services but I do expect I will be able to unravel those mysteries.

Historical side note: DICOM image transfers between many old medical systems do not take place over TCP/IP, as would be normal for the new, IP-based transmission standard. DICOM was once its own transmission protocol, not merely a format encapsulated in a TCP header. I wonder if anyone aside from the original vendors has a way to detect traffic signatures from these old devices. I wouldn't be surprised if some hospitals out there have old, misconfigured networks clogged by legacy DICOM traffic they can't properly diagnose due to poor firewalling, filtering, and/or packet detection.

It is one thing when people try to do reconnaissance and read your SNMP variables, but when they try to set them, that can be fairly significant. Again, never assume you know which systems in your site run SNMP, printer servers, x-terminals, uninterruptible power supplies all have agents.

SNMP WRITE
Sumbitted by Tomas Halvarsson - http://www.sans.org/y2k/practical/Tomas_Halvarsson.txt
Source = Snort logs

```
[**] SNMP-write-write [**]
02/16-22:45:56.398226 0:0:0:55:B6:43 -> 0:0:0:C4:25:8C type:0x800 len:0x5B
0.17.3.44:54064 -> 192.168.1.1:161 UDP TTL:10 TOS:0x0 ID:26881
Len: 14592
30 2F 02 01 00 04 05 77 72 69 74 65 A3 23 02 04  0/.....write.#..
1D 65 1B 74 02 01 00 02 01 00 30 15 30 13 06 0B  .e.t......0.0...
2B 06 01 04 01 84 11 09 05 03 00 40 04 82 11 0E  +..........@....
FE                                               .

[**] SNMP-write-write [**]
02/16-22:45:56.399686 0:0:0:55:B6:43 -> 0:0:0:C4:25:8C type:0x800 len:0x67
0.17.3.44:54064 -> 192.168.1.1:161 UDP TTL:10 TOS:0x0 ID:26882
Len: 17664
30 3B 02 01 00 04 05 77 72 69 74 65 A3 2F 02 04  0;.....write./..
1D 65 1B 75 02 01 00 02 01 00 30 21 30 1F 06 0B  .e.u......0!0...
2B 06 01 04 01 84 11 09 05 04 00 04 10 31 39 32  +.............192
2E 31 36 38 2E 31 2E 31 2E 63 66 67               .168.1.1.cfg

[**] SNMP-write-write [**]
02/16-22:45:56.400959 0:0:0:55:B6:43 -> 0:0:0:C4:25:8C type:0x800 len:0x58
0.17.3.44:54064 -> 192.168.1.1:161 UDP TTL:10 TOS:0x0 ID:26883
Len: 13824
30 2C 02 01 00 04 05 77 72 69 74 65 A3 20 02 04  0,.....write. ..
1D 65 1B 76 02 01 00 02 01 00 30 12 30 10 06 0B  .e.v......0.0...
2B 06 01 04 01 84 11 09 05 01 00 02 01 01        +..............

[**] SNMP-write-write [**]
02/16-22:45:56.402283 0:0:0:55:B6:43 -> 0:0:0:C4:25:8C type:0x800 len:0x5B
0.17.3.44:54064 -> 192.168.1.1:161 UDP TTL:10 TOS:0x0 ID:26884
Len: 14592
30 2F 02 01 00 04 05 77 72 69 74 65 A3 23 02 04  0/.....write.#..
1D 65 1B 77 02 01 00 02 01 00 30 15 30 13 06 0B  .e.w......0.0...
2B 06 01 04 01 84 11 09 05 03 00 40 04 00 00 00  +..........@....
00                                               .

-=-=-=-These four packets, with small variations in the payload-=-=-=-
-=-=-=-depending on the target machine, are sent to 129 other machines,-=-=-=-
-=-=-=-several of which does not exist.-=-=-=-

Analysis:
 The technique:
  The source port is being reused, the packets arrive in a very fast
  succession, and the IDs are incremented by one for each packet. All of this
  indicates that an automated method is being used. Also notice the low TTL: I
```

```
 doubt these packets have been traveling _that_ far, so the sender is probably
 intentionally setting a low TTL when constructing the packets.
The intent:
 This is definitely someone hostile trying to mess with our machines,
 rewriting some snmp info.
The severity:
 Medium. Unfortunately, I don't know much about snmp, but I don't see any
 community string in the packets, so I guess and hope that this attack failed.
 I will however look into it thoroughly.
```

ZONEALARM CATCHES PORT 6970 - GATE CRASHER
Submitted by Gordon Sanborn - http://www.sans.org/y2k/practical/Gordon_Sanborn.doc

Type	Date	Time	Source	Destination: port	Transport
FWIN	4/5/00	21:39:22 -5:00 GMT	209.247.74.56:2179	24.0.xxx.xxx:6970	UDP

IDIC Analysis:

Existence:	Solicited traffic is from 209.247.74.56.
History:	There is no history of this host visiting on a regular basis.
Techniques:	There is only one packet detected from this IP address (200.216.94.66).
Intent:	The intent is to find a server on our subnet that will respond on TCP port 57622.
Targeting:	Probably scanning the entire Internet cable subnet. Unable to verify with only one IP address.
Analysis:	Source address: Nslookup 209.247.74.56 replies with "non-existent domain" Traceroute shows it is one metric hop away from SanDiego.Level3.net. Destination port: 6970/UDP according to www.isi.edu/in-notes/iana/assignments/port-numbers is unassigned. According to www.simovits.com/nyheter9902.html, port 6970/UDP is GateCrasher

Severity:

Component:	Score	Comments
Criticality:	4	My computer is directly targeted
Lethality:	5	GateCrasher is a very lethal. See URL in Detect #5 Analysis
System Counter Measures:	3	Operating system is running the latest patches
Network Counter Measures:	4	Personal Firewall blocks all packets to port 6970/UDP
Severity Score:	2	(Criticality + Lethality) - (System Counter Measures + Network Counter Measures)

Deep Throat is another of the classic Trojans. This originally primarily came out of Russia and eastern Europe and there were several variations of this. If you have a high fidelity logger and you see 2140, be sure to examine the UDP checksum field closely, it is often a representation of the counter on the software trolling from Trojans and increments in reverse to the IP addresses being scanned.

DEEP THROAT SCAN
Submitted by Drew Brunson - http://www.sans.org/y2k/practical/Drew_Brunson.doc

Well this appears to be a down and dirty scan of my network for the Deep Throat trojan from usr11-dialup335.mix1.Irving.cw.net. Since I'm successfully denying the probes, I'm not worried about this. Since the prober doesn't seem to be trying to hide anything I am assuming that the skipped nodes were simply missed. I did let the ISP know this happened. Probably just someone who picked up a script from the Internet.

Mar 30 18:53:12 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.1.2140
Mar 30 18:53:12 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.2.2140
Mar 30 18:53:12 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.3.2140
Mar 30 18:53:12 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.4.2140
(. . . truncated)
Mar 30 18:53:31 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.247.2140
Mar 30 18:53:31 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.248.2140
Mar 30 18:53:32 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.250.2140
Mar 30 18:53:32 fwall 23 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.252.2140
Mar 30 18:53:32 fwall 12 deny: UDP from 166.62.210.89.60000 to xxx.xxx.xxx.255.2140

What is in a ping? What should be? Who would think to check? Analyst Andrew Korty, GCIA does and again, we are all better off for his efforts.

ICMP ECHO REQUESTS
Submitted by Andrew Korty - http://www.sans.org/y2k/practical/Andrew_Korty.txt

This detect looks like a conventional ping scan, and it probably is. However, dumping the packets reveals something unlike any ping packet I've ever seen. Most ping packets contain in their 8-byte header an identifier and a sequence number. Following the header is usually a 56-byte payload, consisting of an 8-byte timestamp and a fill pattern. This FreeBSD ping packet follows the convention:

```
23:14:11.295427 localhost > localhost: icmp: echo request
                        4500 0054 5336 0000 ff01 6a70 7f00 0001
                        7f00 0001 0800 1e15 2263 0000 13ca 0339
                        b181 0400 0809 0a0b 0c0d 0e0f 1011 1213
                        1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
                        2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
                        3435 3637
```

Of course, other ping programs are likely to build different packets, but the ones in this detect do seem odd. The payload is only 12 bytes in length. The identifier is always 0xbeef, which would make it difficult for the sending computer to know which process the replies are intended for, and the sequence number is always 0xdead, so there is no way to tell how many pings have succeeded without being dropped or duplicated. It is interesting to note that 0xdeadbeef is often used by programmers as a fill pattern.

The rest of the payload consists of 6 static bytes and 6 bytes that vary slightly. My initial fear that this is a Loki detect can be dismissed, since there is not enough dynamic information contained in the packets. However, it might be worth further investigation to determine the intent of this attack.

```
06:05:24.834566 195.61.132.6 > 128.210.67.55: icmp: echo request (DF)
                        4500 0028 7873 4000 f101 0614 c33d 8406
                        80d2 4337 0800 656a beef dead 3901 885d
                        0008 6f87 80d2 4337 0000 0000 0000
06:05:24.835882 195.61.132.6 > 128.210.67.74: icmp: echo request (DF)
                        4500 0028 7873 4000 f101 0601 c33d 8406
                        80d2 434a 0800 5928 beef dead 3901 885d
                        0008 7bb6 80d2 434a 0000 0000 0000
06:05:24.844607 195.61.132.6 > 128.210.67.34: icmp: echo request (DF)
                        4500 0028 7869 4000 f101 0633 c33d 8406
                        80d2 4322 0800 740a beef dead 3901 885d
                        0008 60fc 80d2 4322 0000 0000 0000
06:05:24.846490 195.61.132.6 > 128.210.67.186: icmp: echo request (DF)
                        4500 0028 7887 4000 f101 057d c33d 8406
                        80d2 43ba 0800 0f00 beef dead 3901 885d
                        0008 c56e 80d2 43ba 0000 0000 0000
06:05:24.848421 195.61.132.6 > 128.210.67.58: icmp: echo request (DF)
                        4500 0028 7873 4000 f101 0611 c33d 8406
```

```
                        80d2 433a 0800 6375 beef dead 3901 885d
                        0008 7179 80d2 433a 0000 0000 0000
06:06:14.435792 195.61.132.6 > 128.210.67.112: icmp: echo request (DF)
                        4500 0028 3a29 4000 f101 4425 c33d 8406
                        80d2 4370 0800 9f0a beef dead 3901 888f
                        0002 3582 80d2 4370 0000 0000 0000
06:06:14.436990 195.61.132.6 > 128.210.67.117: icmp: echo request (DF)
                        4500 0028 3a29 4000 f101 4420 c33d 8406
                        80d2 4375 0800 9729 beef dead 3901 888f
                        0002 3d5e 80d2 4375 0000 0000 0000
```

The trick here is to do an intro that isn't a spoiler. If you have been going pretty fast, get up, get a cup of coffee or whatever and give this a read from the top down. I know I enjoyed it and I hope you will too.

DNS SCAN
Submitted by Diane Wood - http://www.sans.org/y2k/practical/Diane_Wood.doc

```
Apr 14 03:12:05 : Warning: BADZTREQ: remote host sc1.empire.org/199.74.186.20
      attempted to perform a zone transfer
Apr 14 03:12:05 : Event: EVENTMSG: event badredzt detected from host
      sc1.empire.org/199.74.186.20
Apr 14 03:12:05 : Log: GOTEVENT: event 'dnsxd.badredzt'
      from '199.74.186.20' detected by alarm daemon
Apr 14 03:12:05 : Warning: BADDNSCLASS: invalid DNS request class from host
      sc1.empire.org/199.74.186.20 (class=3)

Apr 14 03:12:10 : Warning: BADZTREQ: remote host sc1.empire.org/199.74.186.20
      attempted to perform a zone transfer
Apr 14 03:12:10 : Event: EVENTMSG: event badredzt detected from host
      sc1.empire.org/199.74.186.20
Apr 14 03:12:10 : Log: GOTEVENT: event 'dnsxd.badredzt'
      from '199.74.186.20' detected by alarm daemon

Apr 14 03:12:10 REJECTN: TCP [199.74.186.20]      [My.Net.34](2366->53)02
Apr 14 03:12:13 REJECTN: TCP [199.74.186.20]      [My.Net.35](2398->53)02
Apr 14 03:12:13 REJECTN: TCP [199.74.186.20]      [My.Net.36](2399->53)02
Apr 14 03:12:13 REJECTN: TCP [199.74.186.20]      [My.Net.37](2400->53)02
Apr 14 03:12:13 REJECTN: TCP [199.74.186.20]      [My.Net.38](2401->53)02
              :                                   :      :
Apr 14 03:12:16 REJECTN: TCP [199.74.186.20]      [My.Net.252](2638->53)02
Apr 14 03:12:16 REJECTN: TCP [199.74.186.20]      [My.Net.253](2639->53)02
Apr 14 03:12:16 REJECTN: TCP [199.74.186.20]      [My.Net.254](2640->53)02
```

Lines deleted for brevity but scan continues in perfect time, source port and destination IP order)

Analysis

Source of Trace: Firewall Logs. The logs reflect rejects based on proxy rules and invalid TCP flags. The TCP flags are shown immediately after the "(source port ->destination port)" in hexadecimal.

Type of Detect/Intent: TCP based attempt to corrupt the DNS and take ownership of the domain.

Active Targeting: Yes, specifically targeting the whole subnetted class C to find and exploit the DNS servers.

History: No previous activity from this IP number or class. Registration information on ARIN displays the address as belonging to a suspiciously named :

The Empire Organization, 2600 Simplex Rd, Goodguess, VA 26001.
The contact address is show as Fracksville,VA.
All hack-like nomenclature/numbers.

Technique: Signature: TCP Script Single source IP number using incremental source ports to a single destination port across the full class C network. *Note: IP numbers 1-33 do not show due to the firewall/router misconfiguration/weakness

but are probably also part of the scan. Unfortuantly, today, I can not see what results were returned for this address range with the existing log/sniffer capabilities.

Fast script sending Syn requests to every machine in our external network. Interesting note was the BADDNSCLASS message generated by the firewall DNS proxy. According to Albitz & Liu in DNS and Bind, O'Reilly, the (class=3) refers to the Chaosnet protocols/class and is historical but infrequently if at all used any more. No other DNS probes I've seen at this site has ever produced this particular error which means it isn't just a SYN scan (which will generate the BADZTREG – bad zone transfer request by the connection request itself).

Severity of Attack/Threat Level: High. Targeted area has DNS servers running within it, not all of which may be up to patch level. DNS administrator was contacted and upgrades were begun. Although maybe this is just script-kiddie work, it is still deliberate activity from a hacker-named site attempting to take control of the DNS servers and it is generating a signature that seems more concerted than the norm.

It isn't that common to see a search for SGI specific ports, but it does happen. This is another great story. Traces and analysis like this make you glad you do intrusion detection, don't they?

UDP PORT 5135
Submitted by Marc Labram - http://www.sans.org/y2k/practical/Marc_Labram.doc

Active Targeting:

Yes

History:

Yes. There have been previous scans from this particular Asian ISP. The System Administrator of goodguy-hacked.com noticed an unauthorized account called zippy on the machine.

Analysis:

This particular trace shows that they were scanning for an open UDP port 5135, the SGI Object Server. As you can see goodguy-a.com and goodguy-b.com answered back with a port unreachable, but open on goodguy-hacked.com

Intent:

The attacker was able to exploit a known hole in SGI's Object Server, which allows a remote user to add a local account.

The following is from the syslog on goodguy-hacked.com:

Apr 15 21:07:59 6C: goodguy-hacked.com telnetd[5020]: connect from some.edu
Apr 15 21:08:18 6E: goodguy-hacked.com login[5021]: ?@some.edu as zippy

Log file:

21:06:26.707866 badguy.com.26614 > goodguy-a.com.5135: udp 52
21:06:26.708960 badguy.com.26614 > goodguy-a.com.5135: udp 52
21:06:26.711804 goodguy-a.com > badguy.com: icmp: goodguy-a.com udp port 5135 unreachable
21:06:26.712748 goodguy-a.com > badguy.com: icmp: goodguy-a.com udp port 5135 unreachable
21:07:01.234448 badguy.com.26616 > goodguy-b.com.5135: udp 52
21:07:01.235528 badguy.com.26616 > goodguy-b.com.5135: udp 52
21:07:01.238380 goodguy-b.com > badguy.com: icmp: goodguy-b.com udp port 5135 unreachable
21:07:01.239430 goodguy-b.com > badguy.com: icmp: goodguy-b.com udp port 5135 unreachable
21:07:16.632941 badguy.com.26617 > goodguy-hacked.com.5135: udp 52
21:07:16.633974 badguy.com.26617 > goodguy-hacked.com.5135: udp 52
21:07:16.668785 goodguy-hacked.com.5135 > badguy.com.26617: udp 69
21:07:16.669475 goodguy-hacked.com.5135 > badguy.com.26617: udp 69
21:07:16.897641 badguy.com.26617 > goodguy-hacked.com.5135: udp 308
21:07:16.898367 badguy.com.26617 > goodguy-hacked.com.5135: udp 308
21:07:17.838778 goodguy-hacked.com.5135 > badguy.com.26617: udp 41
21:07:17.839684 goodguy-hacked.com.5135 > badguy.com.26617: udp 41

If you read GIAC, you will see Binette's firewall reports from the home.com network in almost every issue. His reports are a valuable indication of just exactly how crazy things are in cable modem land.

TROJAN SCANS
Submitted by Timothy Trow - http://www.sans.org/y2k/practical/Timothy_Trow.doc
Source = http://www.sans.org/y2k/032800-2000.htm

From an @home user… smattering of Netbus, a pinch of BackOrifice, a bit of SubSeven…

> Mar 27 00:09:52 cc1014244-a kernel: securityalert: tcp if=ef0 from 209.235.11.254:50998 to 24.3.21.199 on unserved port 512
> Mar 27 01:08:04 cc1014244-a kernel: securityalert: tcp if=ef0 from 24.40.35.215:2790 to 24.3.21.199 on unserved port 12345
> Mar 27 08:18:19 cc1014244-a kernel: securityalert: tcp if=ef0 from 38.27.95.44:2756 to 24.3.21.199 on unserved port 1080
>
> Mar 27 19:22:03 cc1014244-a kernel: securityalert: tcp if=ef0 from 38.26.9.97:4882 to 24.3.21.199 on unserved port 1080
> Mar 27 20:29:59 cc1014244-a kernel: securityalert: tcp if=ef0 from 151.198.141.96:2660 to 24.3.21.199 on unserved port 27374
>
> Mar 27 20:50:25 cc1014244-a kernel: securityalert: tcp if=ef0 from 151.198.141.96:2344 to 24.3.21.199 on unserved port 27374
>
> Mar 27 20:57:48 cc1014244-a kernel: securityalert: tcp if=ef0 from 207.50.63.40:3537 to 24.3.21.199 on unserved port 1243
>
> Mar 27 22:07:59 cc1014244-a kernel: securityalert: udp if=ef0 from 38.32.11.6:1044 to 24.3.21.199 on unserved port 31337

Active Targeting?

Yes. The traffic is being targeted towards the same address on various ports.

History:

No previous history was noted in the detect report.

Technique(s):

All activity was within one day, March 27. The times were very different. Times ranged from 10 minutes past midnight to later that evening. The attacker seemed to be very careful about keeping the times at a distance. The source IP is multiple IP's, or someone is spoofing those IP's. The attacker is targetting the same destination address on every attempt, but targeting various destination ports. The various random source addresses really scare me. This is not a pretty picture. This could be some type of script running or a very careful and collective attack methodology!

Analysis/Intent:

There are many possible exploits being targeted here. Port 512/tcp is UNIX remote exec. This may be a way trying to gain access to a UNIX system by issuing remote commands. Rexecd allows redirection of stderr stream to an arbitrary port on the client machine. This stream is opened by rexecd before authentication of the user. Spoofing techniques could allow the client to direct the stderr stream towards an arbitrary host as well as an arbitrary port, possibly exploiting a given trust model. Another port of interest is port 12345/tcp. Netbus is similar

to BackOrifice. It allows ANYONE running the client portion to connect and control ANYONE running the server portion of it, WITH THE SAME RIGHTS AND PRIVILEGES AS THE CURRENTLY LOGGED ON USER! It allows the remote user total access. The next port 1080/tcp is a SOCKS proxy port. This could be used as a potential spam relay point. Ports 27374/tcp are examples of sub-7 (or also known as subseven). This is a default port that appeared in v2.0. This port along with port 1243/tcp are very well known trojan ports. I found the following on this: Supports "port redirection", so that any attack can be funneled through a victim's machines. Contains extensive tricks to play with ICQ, AOL IM, MSN Messenger, and Yahoo messenger, including password sniffing, posting messages, and other features. Extensive UI tricks, such as flipping the screen, talking through the victim's speaker, and spying on the victim's screen. Sub7 is written by a hacker who calls himself "Mobman". The last trace show destination port 31337/tcp being targeted. This is the well known trojan horse port called BackOriffice. It is similar in action to the Netbus trojan horse, very dangerous and common among the hacker community. The targeted **port scanning** of these well-known ports is indicative of a need to exploit. I would be very concerned with these "attempts". What seems to be the saving grace is just that, they all seemed to be failed attempts. The **cc1014244-a kernel: securityalert:** seems to indicate that these attempts had failed. This seems to be a UNIX platform running some type of Firewall software.

Identify Hostile Individuals and Groups?

It may be hard to track these down. NOTE: I attempted an http to 209.235.11.254 (http://209.235.11.254/) and found that it brought me to the web page for "The Digital LandLords". Now it gets a little strange from here I have to confess. I clicked on the link named, The Digital LandLords and it brought me to the following URL; http://home.clever.net/. The web page described Clever Internet Services, A Division of Interliant. What is so strange about this is that I work for a company in Woburn. MA called Triumph Technologies, Inc. Interliant bought us up in November 1999. Interliant is in the ASP, Web hosting business...so I have a feeling that the would be attacker may be using their web servers as a launching pad for their attacks. Since Clever.net is an ISP located in Atlanta, this may be an inside job so to speak. An ISP/Web hosting facility would be able to gain access to multiple of servers to launch prospective attacks. The times also warrant this. They span the whole night!

Components:	**Score:**	**Comments:**
Criticality:	*5*	*Attacks are against one machine.*
Lethality:	*5*	*These ports of attack are very lethal and some are well-known trojan horses*
System Countermeasures:	*4*	*The system seems to be blocking these ports*
Network Countermeasures:	4	The network does a decent job of blocking these ports.
Severity Total:	2	**(Criticality + Lethality) – (System Countermeasures + Network countermeasures)**

Here is a great analysis of another load balancing pattern. I really like this submission because Rob Ryan, GCIA gives us a real feeling for the thought process he went through to classify these firewall detects.

LOAD BALANCING
Submitted by Ron Ryan - http://www.sans.org/y2k/practical/Ron_Ryan.doc

On March 29, starting at 00:14.45, the preceding detects were logged on one of my customer's firewall . There are actually hundreds more of these entries that occurred over the course of the day. Each packet is a UDP packet with a length of 64, a destination port of 33434 and is destined for one of three nodes on the site. My.net.100.100 is a gateway. My.net.76.5 and .91.5 are Windows NT Proxy servers acting as cache servers. The firewall dropped all of the packets and logged them appropriately.

This particular signature confused me at first because of the consistent use of destination port 33434. Port 33434 is the first port used in a UDP trace route - see the IANA port assignments located at: http://www.isi.edu/innotes/iana/assignments/port-numbers. Normally when a trace route is performed the destination port number and TTL (Time to Live) values are incremented on each attempt. The first packet sent will use destination port 33434 and have a TTL of 1. The first router that receives the packet decrements the TTL value to 0, which causes an ICMP time exceeded error. The second packet sent by traceroute will use a TTL value of 2 and a destination port of 33435. This activity will continue with the TTL value being incremented by trace route for each hop attempt and decremented at each intervening router. The destination port is as well, normally incremented on each attempt. The port number is normally a good indicator of how far away the tracing host is, assuming you don't have the TTL value. The larger the value, the more hops that have taken place. You won't normally see a traceroute packet with a port value of 33434 that gets to your firewall. Intervening routers will have caused trace route to increment this value on each successive attempt.

In this case I do not have a TTL value for the packet as this is a firewall log and greater detail was not available. I can only surmise that this value is being incremented appropriately by the load balancer on each attempt, as the source addresses are numerous hops away from the destination hosts. There is another reason for making this ***assumption*** and that revolves around the particular activity we are seeing here.

Trace #3 – Firewall Log

Number	Date and Time	Act	port	Source	Destination	Prot	Length
"56"	"29Mar2000"	" 0:14:45"	"drop"	"33434"	"206.191.171.11"	"my.net.76.5"	"udp" " len 64"
"57"	"29Mar2000"	" 0:14:45"	"drop"	"33434"	"206.191.171.11"	"my.net.76.5"	"udp" " len 64"
"58"	"29Mar2000"	" 0:14:45"	"drop"	"33434"	"206.191.171.11"	"my.net.76.5"	"udp" " len 64"
"3608"	"29Mar2000"	" 9:29:29"	"drop"	"33434"	"209.67.123.169"	"my.net.76.5"	"udp" " len 64"
"3609"	"29Mar2000"	" 9:29:29"	"drop"	"33434"	"209.67.123.169"	"my.net.76.5"	"udp" " len 64"
"3610"	"29Mar2000"	" 9:29:29"	"drop"	"33434"	"209.67.123.169"	"my.net.76.5"	"udp" " len 64"
"3620"	"29Mar2000"	" 9:31:13"	"drop"	"33434"	"209.67.123.169"	"my.net.91.5"	"udp" " len 64"
"3621"	"29Mar2000"	" 9:31:13"	"drop"	"33434"	"209.67.123.169"	"my.net.91.5"	"udp" " len 64"
"3622"	"29Mar2000"	" 9:31:13"	"drop"	"33434"	"209.67.123.169"	"my.net.91.5"	"udp" " len 64"
"5244"	"29Mar2000"	"11:24:49"	"drop"	"33434"	"206.191.171.4"	"my.net.91.5"	"udp" " len 64"
"5245"	"29Mar2000"	"11:24:49"	"drop"	"33434"	"206.191.171.4"	"my.net.91.5	"udp" " len 64"
"5246"	"29Mar2000"	"11:24:49"	"drop"	"33434"	"206.191.171.4"	"my.net.91.5"	"udp" " len 64"
"7215"	"29Mar2000"	"12:51:30"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp" " len 64"
"7216"	"29Mar2000"	"12:51:31"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp" " len 64"
"7219"	"29Mar2000"	"12:51:32"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp" " len 64"
"7221"	"29Mar2000"	"12:51:33"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp" " len 64"
"7225"	"29Mar2000"	"12:52:27"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp" " len 64"
"7226"	"29Mar2000"	"12:52:28"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp" " len 64"
"7227"	"29Mar2000"	"12:52:29"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp" " len 64"
"7228"	"29Mar2000"	"12:52:30"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp" " len 64"
"7229"	"29Mar2000"	"12:52:31"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp" " len 64"
"7231"	"29Mar2000"	"12:54:43"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp" " len 64"
"7232"	"29Mar2000"	"12:54:44"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp" " len 64"
"7233"	"29Mar2000"	"12:54:46"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp" " len 64"
"7234"	"29Mar2000"	"12:54:47"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp" " len 64"
"7247"	"29Mar2000"	"12:57:22"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7248"	"29Mar2000"	"12:57:23"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7249"	"29Mar2000"	"12:57:24"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7250"	"29Mar2000"	"12:57:25"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7251"	"29Mar2000"	"12:57:25"	"drop"	"33434"	"206.251.19.89"	"my.net.100.100"	"udp" " len 64"
"7252"	"29Mar2000"	"12:58:31"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7253"	"29Mar2000"	"12:58:32"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7254"	"29Mar2000"	"12:58:33"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7255"	"29Mar2000"	"12:58:34"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7257"	"29Mar2000"	"12:59:46"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7258"	"29Mar2000"	"12:59:47"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7259"	"29Mar2000"	"12:59:49"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"
"7260"	"29Mar2000"	"12:59:50"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp" " len 64"

Analysis:

The groups responsible for these detects are:

206.191.171.11	Exodus Communications - Exodus.net
209.67.123.169	Rival Communications Network - www.rivalcom.net
206.191.171.4	Rival Communications Network
206.251.19.88	Global Crossing - telecommunications company - www.globalcrossing.com
167.8.29.91	Gannett Company - News and information Company - www.gannett.com
209.67.29.8	Exodus Communications - Exodus.net - USAToday - www.usatoday.com
209.67.29.9	Exodus Communications - Exodus.net- USAToday - www.usatoday.com

This activity is caused by WEB browsing though the stimulus is not shown in the log. Local users are browsing a number of different NEWS services that have Web Servers load balanced across multiple sites. The traffic we are seeing is the result of load balancers that are using a hop count algorithm to determine the closest application server to the requesting client.

There are a number of products on the market today, from companies like Cisco, Resonate, Nortel, F5LABS and GTE, that perform distributed load balancing. F5Labs' 3DNS, GTE Hopscotch, Resonate Global Dispatch, and Cisco's Distributed Director are all capable of performing load balancing across multiple Points of Presence or POPs. There are as well numerous algorithms that may be used by these products to determine the best path for a customer; DNS caching name server, HTTP session director, E-Business rules, SNMP metric probing, round robin, and of course hop count.

The hop count mechanism is a method of mapping the best route from the client to a POP. The latency, as measured by the trace route mechanism used by the load balancer, establishes an optimal path. The trace is initiated when a gethostbyname query is issued on behalf of the client by the local DNS server or cacheing appliance. The trace is directed at the source of the query and not at the client.

Resonate has a good description of the process flow on their web site which I have included here:

When a client connects to an internet site, it must resolve the site name (e.g., www.resonate.com) to an IP address.

The client sends a request to the local DNS.
If the requested site information is not cached in the local DNS, the local DNS server sends a request to the requested site's authoritative DNS server. The DNS server examines the name requested and determines the actual subdomain.
Global Dispatch determines if the scheduling information has been defined through an Advanced Traffic Mapping rule, or stored in cache.
If the information is not already defined or available in cache, Global Dispatch evaluates site load availability, computes latency, and determines the most appropriate POP for the client.
The IP address of the appropriate POP is returned to the client.
Client traffic is directed to the POP.

There are a number of products I know that uses a hop count algorithm, however this fingerprint is the F5LABS 3DNS controller. I have installed load balancers before at E-Commerce sites and I have run into F5Labs before. More info can be found on this product at: http://www.f5labs.com/solutions/whitepapers/3dnscontroller.html

The following is the correspondence I had with F5Labs about this issue:

My query:

I have a number of intrusion detects on customer's sites that appear to be traceroute mapping of the customer's site. The footprint for these is a UDP destination port of 33434 (the first port of traceroute using udp). Several attempts are made with the destination port not incrementing which is the usual behavior of traceroute. In none of these detects do I have the TTL value so I have to assume that it is the actual metric that is incremented by the load balancer and decremented by intervening nodes. All of this activity appears to be the result of WEB queries by users with a response from a distributed load balancer that uses a hop count algorithn. Can you tell me if this is in deed the behavior of your 3dns product when the hop count algorithm is in use?

F5Labs response:
This is correct. The only destination you should see in these queries are the dns server that queried the 3dns box. After recieveing a request the 3dns will respond with a default address for the client to go to, and then start to probe the dns server that the request came from so future requests from that dns server can be answered with the best site, according to the criteria defined in the configuration. The 3dns will continue probing the dns box long after the client is done at the site the 3dns is serving.

They state that only the DNS server should be receiving these packets. In this case the Proxy caching servers and the gateway, an Alpha Unix DNS server, are challenging the DNS servers on these sites and this is generating the hop count pattern we are seeing to these servers.

The reason for this is spelled out in RFC 1919.

“In most classical-proxy configurations, client systems pass the desired server name (or address) to the proxy system WITHOUT INTERPRETING IT. Because of this, the client system DOES NOT REQUIRE to be able to resolve the name of the server system in order to access it through a classical proxy. It only needs to be able to resolve the name of the proxy (if referencing the proxy system by name). Because of this, it can be said that a classical proxy system can offer DNS isolation. If two IP internetworks use completely separate DNS trees (each with their own DNS root servers), client software in one IP internetwork may still reference a server name in the other IP internetwork by passing its name to the classical proxy. The classical proxy itself will not be able alone to resolve DNS names in both environments (if running standard DNS resolution software), since it will need to point to one or the other of the two DNS "universes". “

Also, “Classical proxy connections have no impact on normal server software; the proxy looks like a normal client in most respects except for its IP address and its "group" nature. All connections from the network on the other side of the proxy appear to come from the proxy.”

This basically means that the Windows NT proxy server--cache server—becomes the client and does the gethostbyname query for the client. The result is that the load balancer directs it’s hop count traffic to the proxy.

Note that F5Labs states that the 3DNS will continue to probe the DNS box long after the client is done at the site. So you may see this traffic long after everyone has gone home for the evening.

I decided to contact one of the sites that was generating this traffic and chose Gannett Co. because I didn't know who they were. This is a response I got back from Gannett Co. All of the sites listed above are co-located web servers.

> "Ron,
> What you are seeing in your logs are load-balancing packets because your customers are hitting http://www.usatoday.com. When a user or proxy server requests a lookup for http://www.usatoday.com, they are redirected to the first usatoday.com name server. After that first request, our load-balancing systems perform checks to determine which of our topographically closest co-location facilities to direct future traffic from that address to. The systems use pings, dns queries, and traceroutes to determine best round trip times, reliability checks, etc.
>
> You should be able to corroborate the times of the packets with any proxy logs you may have in order to verify the occurances. Although the logs you have provided do not indicate dns rule violations, if you log outbound DNS queries, you'll probably see a closer match in timings.
>
> You are right, they are F5 boxes.
>
> - Sam

This is targeted traffic but perhaps not in the normal sense because it is the result of user browsing activity. This therefore, is not hostile traffic, as the Firewall manager that sent me these logs and I initially thought, but nuisance traffic. In this particular case the firewall is dropping the packets, so "no harm, no foul.

Active Targeting: Yes

History: This is ongoing activity.

Technique: Use of a simple trace route algorithm to find the shortest path to the DNS server or proxy server that performed a gethostbyname query.

Intent: Find the shortest path between the initiating web browser's site and the web sites that the load balancer is responsible for.

Severity/Criticality: This is targeted at Core infrastructure systems. DNS servers and proxy servers. **5**

Severity/Lethality: 1 This is a mapping technique not an attack.

System Countermeasures: 3 Unknown but not hardened systems.

Network Countermeasures: 4 Restrictive firewall that is doing it's job.

Severity = (5+1) - (3+4) = **-1 Very low.**
Note: There are 2 things of note.

1) If we know about this type of load balancing and it's unique signature, you can bet that the "bad guys" know about it too. There are tools available that allow the mapper to modify the initial port used, number of attempts, whether or not the destination port increments and the source port. In particular, a technique known as firewalking can be used to map your point(s) of ingress and provide intimate knowledge of the firewall rules that you have in place. Once initial mapping using trace route is

performed, the firewall can then be mapped using different source ports with queries directed to hosts behind the firewall. To detect this type of technique you need to look at the source port, the destination port, and the target function and port they are attempting to probe. A common source port for a probe using UDP would be port 53.

2) If you want to insure that this activity is indeed load balancing then take a look at the caching appliance or DNS server's outbound DNS logs. This can help in correlating the web site accessed with the load balancing response that you receive. Unsolicited traffic where there is no correlation of outbound DNS logs and the trace activity could be actual mapping attempts.

Another great job of an analyst showing how they think. I have been tracking the PCAnywhere pattern since the beginning of GIAC and it is amazing how often this shows up!

PC ANYWHERE PROBES
Submitted by Fred Kolbrener - http://www.sans.org/y2k/practical/Fred_Kolbrener.doc

PCAnywhere probes. The first event occurred on April 1, 2000 just after 12 Noon. BlackICE Defender (BID) reported an intrusion attempt as shown below. The time shown as posted to the attack log is GMT or UFT time which was 5 hours ahead of Eastern Standard Time. The data was originally recorded as a comma delimited ASCII file that was subsequently imported into Excel for analysis. An explanation of the entry is as follows:

Severity as assigned by BID: 19 (Scale runs from 1 to 99)
Date/Time (GMT) [YYYY-MM-DD hh:mm:ss]: 2000-04-01 17:14:08
Event Type (BID Code): 2001507
Event Name: PCAnywhere ping
IP Address of source computer: 207.172.73.49
NetBIOS name of source computer: ROADRUNNER
IP of Target computer: 207.172.73.41
Port(s) on targeted computer: 22 & 5632
Number of "sub-events": 2

19 2000-04-01 17:14:08 2001507 PCAnywhere ping 207.172.73.49 207.172.73.41 port=22|5632 2

Concurrently, this attacked computer was running WINDUMP (version 2.02 beta) and was recording standard length (68) byte packet headers of all incoming and outgoing traffic. (No filtering of events was in place at that time.) The WINDUMP output file was queried for all events with a source computer of 207.172.73.49. The following list was produced:

12:11:47.311800 207.172.73.49.3123 > 207.172.73.41.5632: udp 2
12:11:47.321198 207.172.73.49.3123 > 207.172.73.41.22: udp 2
12:11:48.520847 207.172.73.49.137 > 207.172.73.41.1598: udp 283

Reviewing the first two lines of data, we see one side of the conversation, receipt of data. The first two lines indicate that the prober is using his ethereal port 3123 to send pings probes for computers running PC anywhere (port 22 or 5632). The fact that the source port does not increment and the probes are very fast supports the use of an unidentified tool to do this probing.

Listing showing MAC addresses (dial-up modem) and size of the UDP packets:

12:11:47.311800 20:53:52:43:0:0 44:45:53:54:0:0 ip 44: 207.172.73.49.3123 207.172.73.41.5632: udp 2
12:11:47.321198 20:53:52:43:0:0 44:45:53:54:0:0 ip 44: 207.172.73.49.3123 > 207.172.73.41.22: udp 2
12:11:48.520847 20:53:52:43:0:0 44:45:53:54:0:0 ip 325: 207.172.73.49.137 > 207.172.73.41.1598: udp 283

The third line is a response to a NetBIOS request launched by BID to which the prober's computer answered with the computer name, and other information. This shows the prober was assigned an address by DHCP on the same network dial-up segment as the probed computer (207.172.72).

IP: 207.172.73.49
DNS: 207-172-73-49.s49.tnt1.man.va.dialup.rcn.com
Node: ROADRUNNER
NetBIOS: SMELLY
Group: ACME

HEX DUMP from WINdump confirms the probe in UDP packets (protocol 0x11):

```
12:11:47.311800 207.172.73.49.3123 > 207.172.73.41.5632: udp 2
                        4500 001e 8b7d 0000 7f11 7e9e cfac 4931
                        cfac 4929 0c33 1600 000a 5da2 4e51
12:11:47.321198 207.172.73.49.3123 > 207.172.73.41.22: udp 2
                        4500 001e 8c7d 0000 7f11 7d9e cfac 4931
                        cfac 4929 0c33 0016 000a 738c 4e51
12:11:48.520847 207.172.73.49.137 > 207.172.73.41.1598: udp 283
                        4500 0137 c97d 0000 7f11 3f85 cfac 4931
                        cfac 4929 0089 063e 0123 4a4d 80b0 8400
                        0000 0001 0000 0000 2043 4b41 4141 4141
                        4141 4141 4141
```

On Sunday, April 2, a second event was recorded as below (from a computer named DEFAULT). Two pings of port 22 were recorded. The source was a different computer as evidenced by the different NetBIOS name and also the different mode of the ping. Only port 22 was targeted.

19 2000-04-03 01:10:25 2001507 PCAnywhere ping 207.172.73.166 DEFAULT 207.172.73.134 port=22 2

Since these initial pings, additional pings apparently from other computers have been recorded. Note that the NetBIOS names are different and there is a lack of consistency in the probing. Sometimes there is a single port probed and sometimes two are probed. From reading about PCAnywhere, I have found that it apparently uses port 22 for initial contact, and may then switch to port 5632 (default) for data transfer, although it can be set to use any port the user desires.

19 2000-04-15 20:36:30 2001507 PCAnywhere ping 207.172.73.127 207-172-73-127.s127.tnt1.
man.va.dialup.rcn.com 207.172.73.188 port= 22|5632 2
19 2000-04-16 03:55:21 2001507 PCAnywhere ping 207.172.73.108
FASTCOMPUTER 207.172.73.187 port= 22|5632 2

ANALYSIS:

Was I targeted? Yes, the IP address was targeted by a probe; however, since the ISP's DHCP assigns an address each time we establish a connection to the ISP, this particular computer was not ***specifically*** targeted.
Were we at risk? NO, this particular computer 1) does not run PCAnywhere, and 2) sits behind a firewall that filters out attacks

Were the probes hostile: Yes. There is and was no valid reason for these events to have occurred. Interesting also is the fact that this probing is taking place on my ISP's own network (207.172.73.XXX) from another ISP user, not from an outside source. The intrusions have all been reported to the ISP. One possible explanation still needs to be followed up. It has been proposed that PCAnywhere , when installed on a computer, may ping the local network looking for other copies of itself each time it is able to connect to a network. Testing to this hypothesis is planned for the future when I can get the program to test with.

This trace is just a bit long, but it is good to document the behavior of large scale scans. I agree with the analyst, there is a good probability this is the result of nmap.

FULL BORE PORT SCAN
Submitted by Brian Betterton - http://www.sans.org/y2k/practical/brian_betterton.doc

20:07:58.111221 212.204.216.151.63118 > my.firewall.net.5300: . win 2048
20:07:58.112097 212.204.216.151.63118 > my.firewall.net.2038: . win 2048
20:07:58.112967 212.204.216.151.63118 > my.firewall.net.348: . win 2048
20:07:58.113843 212.204.216.151.63118 > my.firewall.net.915: . win 2048
20:07:58.114722 212.204.216.151.63118 > my.firewall.net.798: . win 2048
20:07:58.115603 212.204.216.151.63118 > my.firewall.net.425: . win 2048
20:07:58.116477 212.204.216.151.63118 > my.firewall.net.1352: . win 2048
20:07:58.117395 212.204.216.151.63118 > my.firewall.net.2605: . win 2048
20:07:58.118279 212.204.216.151.63118 > my.firewall.net.308: . win 2048
20:07:58.119163 212.204.216.151.63118 > my.firewall.net.859: . win 2048
20:07:58.120080 212.204.216.151.63118 > my.firewall.net.7006: . win 2048
20:07:58.120994 212.204.216.151.63118 > my.firewall.net.1520: . win 2048
20:07:58.121871 212.204.216.151.63118 > my.firewall.net.535: . win 2048
20:07:58.122747 212.204.216.151.63118 > my.firewall.net.5193: . win 2048
20:07:58.123627 212.204.216.151.63118 > my.firewall.net.383: . win 2048
20:07:58.124510 212.204.216.151.63118 > my.firewall.net.1477: . win 2048
20:07:58.125384 212.204.216.151.63118 > my.firewall.net.443: . win 2048
20:07:58.126254 212.204.216.151.63118 > my.firewall.net.9876: . win 2048
20:07:58.127132 212.204.216.151.63118 > my.firewall.net.530: . win 2048
20:07:58.128004 212.204.216.151.63118 > my.firewall.net.659: . win 2048
20:07:58.128934 212.204.216.151.63118 > my.firewall.net.41: . win 2048
20:07:58.129815 212.204.216.151.63118 > my.firewall.net.206: . win 2048
20:07:58.130818 212.204.216.151.63118 > my.firewall.net.371: . win 2048
20:07:58.131690 212.204.216.151.63118 > my.firewall.net.2003: . win 2048
20:07:58.132570 212.204.216.151.63118 > my.firewall.net.1016: . win 2048
20:07:58.133456 212.204.216.151.63118 > my.firewall.net.510: . win 2048
20:07:58.134342 212.204.216.151.63118 > my.firewall.net.567: . win 2048
20:07:58.167773 my.firewall.net.5300 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.168675 my.firewall.net.2038 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.169581 my.firewall.net.348 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.170449 my.firewall.net.915 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.171338 my.firewall.net.798 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.172234 my.firewall.net.425 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.173130 my.firewall.net.1352 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.174016 my.firewall.net.2605 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.174979 my.firewall.net.308 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.175862 my.firewall.net.859 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.176737 my.firewall.net.7006 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.177628 my.firewall.net.1520 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.178537 my.firewall.net.535 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.179412 my.firewall.net.5193 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.180304 my.firewall.net.383 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.181193 my.firewall.net.1477 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.182090 my.firewall.net.443 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.182982 my.firewall.net.9876 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.183862 my.firewall.net.530 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.184790 my.firewall.net.659 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0

20:07:58.185645 my.firewall.net.41 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.186549 my.firewall.net.206 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.187447 my.firewall.net.371 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.188334 my.firewall.net.2003 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.189223 my.firewall.net.1016 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0
20:07:58.190166 my.firewall.net.510 > 212.204.216.151.63118: R 0:0(0) ack 0 win 0

additional trace skipped…then we see this interesting activity on ports 80 and 1354…

20:07:59.991914 212.204.216.151.63118 > my.firewall.net.80: . win 2048
20:08:00.370883 212.204.216.151.63119 > my.firewall.net.80: . win 2048
20:08:00.796363 212.204.216.151.63125 > my.firewall.net.80: SE 3102347426:3102347426(0) win 2048 <wscale 10,nop,mss 265,timestamp 1061109567[|tcp]>
20:08:00.797243 212.204.216.151.63126 > my.firewall.net.80: . win 2048 <wscale 10,nop,mss 265,timestamp 1061109567[|tcp]>
20:08:00.798097 212.204.216.151.63127 > my.firewall.net.80: SFP 3102347426:3102347426(0) win 2048 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567[|tcp]>
20:08:00.798948 212.204.216.151.63128 > my.firewall.net.80: . ack 0 win 2048 <wscale 10,nop,mss 265,timestamp 1061109567[|tcp]>
20:08:00.799806 212.204.216.151.63129 > my.firewall.net.1354: S 3102347426:3102347426(0) win 2048 <wscale 10,nop,mss 265,timestamp 1061109567[|tcp]>
20:08:00.800761 212.204.216.151.63130 > my.firewall.net.1354: . ack 0 win 2048 <wscale 10,nop,mss 265,timestamp 1061109567[|tcp]>
20:08:00.801614 212.204.216.151.63131 > my.firewall.net.1354: FP 3102347426:3102347426(0) win 2048 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567[|tcp]>
20:08:00.802640 212.204.216.151.63118 > my.firewall.net.1354: udp 300
20:08:00.810505 my.firewall.net.80 > 212.204.216.151.63125: S 1348112897:1348112897(0) ack 3102347427 win 4096 <mss 1460>
20:08:00.811122 212.204.216.151.63125 > my.firewall.net.80: R 3102347427:3102347427(0) win 0
20:08:00.813452 my.firewall.net.80 > 212.204.216.151.63127: . ack 3102347428 win 4096
20:08:00.814081 212.204.216.151.63127 > my.firewall.net.80: R 3102347428:3102347428(0) win 0
20:08:00.814963 my.firewall.net.80 > 212.204.216.151.63128: R 0:0(0) win 4096
20:08:00.816255 my.firewall.net.1354 > 212.204.216.151.63129: R 0:0(0) ack 3102347427 win 0
20:08:00.817227 my.firewall.net.1354 > 212.204.216.151.63130: R 0:0(0) win 0
20:08:00.818227 my.firewall.net.1354 > 212.204.216.151.63131: R 0:0(0) ack 3102347426 win 0
20:08:00.819175 my.firewall.net > 212.204.216.151: icmp: my.firewall.net udp port 1354 unreachable
20:08:01.172519 212.204.216.151.63119 > my.firewall.net.80: S 3102347427:3102347427(0) win 2048
20:08:01.177209 my.firewall.net.80 > 212.204.216.151.63119: S 1348240897:1348240897(0) ack 3102347428 win 4096 <mss 1460>
20:08:01.177847 212.204.216.151.63119 > my.firewall.net.80: R 3102347428:3102347428(0) win 0
20:08:01.250806 212.204.216.151.63120 > my.firewall.net.80: S 3102347428:3102347428(0) win 2048
20:08:01.255225 my.firewall.net.80 > 212.204.216.151.63120: S 1348304897:1348304897(0) ack 3102347429 win 4096 <mss 1460>
20:08:01.255880 212.204.216.151.63120 > my.firewall.net.80: R 3102347429:3102347429(0) win 0
20:08:01.330876 212.204.216.151.63121 > my.firewall.net.80: S 3102347429:3102347429(0) win 2048
20:08:01.335256 my.firewall.net.80 > 212.204.216.151.63121: S 1348432897:1348432897(0) ack 3102347430 win 4096 <mss 1460>
20:08:01.335904 212.204.216.151.63121 > my.firewall.net.80: R 3102347430:3102347430(0) win 0
20:08:01.410846 212.204.216.151.63122 > my.firewall.net.80: S 3102347430:3102347430(0) win 2048
20:08:01.415248 my.firewall.net.80 > 212.204.216.151.63122: S 1348496897:1348496897(0) ack 3102347431 win 4096 <mss 1460>
20:08:01.415904 212.204.216.151.63122 > my.firewall.net.80: R 3102347431:3102347431(0) win 0
20:08:01.490832 212.204.216.151.63123 > my.firewall.net.80: S 3102347431:3102347431(0) win 2048
20:08:01.495271 my.firewall.net.80 > 212.204.216.151.63123: S 1348560897:1348560897(0) ack 3102347432 win 4096 <mss 1460>
20:08:01.495931 212.204.216.151.63123 > my.firewall.net.80: R 3102347432:3102347432(0) win 0
20:08:01.574627 212.204.216.151.63124 > my.firewall.net.80: S 3102347432:3102347432(0) win 2048
20:08:01.579168 my.firewall.net.80 > 212.204.216.151.63124: S 1348624897:1348624897(0) ack 3102347433 win 4096 <mss 1460>
20:08:01.579784 212.204.216.151.63124 > my.firewall.net.80: R 3102347433:3102347433(0) win 0

Existence: 212.204.216.151 is "hosted-by.widexs.nl"

```
European Regional Internet Registry/RIPE NCC (NET-RIPE-NCC-)
        These addresses have been further assigned to European users.
        Contact information can be found in the RIPE database, via the
        WHOIS and TELNET servers at whois.ripe.net, and at
        http://www.ripe.net/db/whois.html

        Netname: RIPE-NCC-212
        Netblock: 212.0.0.0 - 212.255.255.255
        Maintainer: RIPE

        Coordinator:
           RIPE Network Coordination Centre  (RIPE-NCC-ARIN)  nicdb@RIPE.NET
           +31 20 535 4444
     Fax- - +31 20 535 4445

        Domain System inverse mapping provided by:

        NS.RIPE.NET                   193.0.0.193
        NS.EU.NET                     192.16.202.11
        AUTH03.NS.UU.NET              198.6.1.83
        NS2.NIC.FR                    192.93.0.4
        SUNIC.SUNET.SE                192.36.125.2
        MUNNARI.OZ.AU                 128.250.1.21
        NS.APNIC.NET                  203.37.255.97

        To search on arbitrary strings, see the Database page on
        the RIPE NCC web-site at http://www.ripe.net/db/

        Record last updated on 16-Oct-1998.
        Database last updated on 24-Apr-2000 17:39:54 EDT.
```

History: **None previously observed.**

Techniques: This was a fast host port scan. Notice the same source port number 63118 during the majority of the scan. They also have used a null scan. A null scan turns off all flags. This type of scan is intended to go un-noticed by firewalls and packet filters looking for SYNs, and programs like Synlogger and Courtney. The scanner scanned over 1500 ports before scanning ports 80 and 1354. Notice that my firewall replied to these closed ports with a RST, as expected (refer to RFC 973).
The source ports then start incrementing as it begins to scan ports 80 and 1354. The scanning tool is probably nmap, as it appears that the further scan activity on ports 80 and 1354 may have been an effort at OS identification, using TCP/IP fingerprinting.

Targeting: Absolutely! This source has targeted my firewall specifically, scanning over 1500 ports and then trying to do an OS identification.

Analysis: The intruder is trying to determine the host type of my firewall and what open ports exist. Their next step might include trying specific known vulnerabilities against my firewall, assuming they can determine the type of system and what weaknesses it has.

Severity Level: (Critical + Lethal) – (System Countermeasures + Net Countermeasures) = Severity Level

This is a firewall (5). They may have identified my firewall and have definitely targeted me, so I'm giving this a (3). They may be aware of vulnerabilities in this firewall that I'm not aware of.
HTTP was temporarily allowed through the firewall. (Time to turn this back off). The internal systems and network countermeasures are above average, but due to the unknown, I'll average the countermeasures. This may not rate a high severity level, but its worth performing some testing on the firewall myself to see what else they may have found.

$$(5 + 3) – (3 + 3) = 2$$

Solution: Turn off allowing HTTP on firewall…this was a temporary rule for a test. Research firewall vendor's bug list to verify it is up-to-date.

Another example of a CGI probe or attack, but in this case it is being used as a port scan. The curious aspect to this trace and analysis is that you could only detect the cgi-bin attack after the three way handshake is complete meaning the systems in the first list are web servers. Viriya works for an ISP if I remember correctly so that would explain that.

"TEST-CGI" WEB ATTACK
Submitted by Viriya Upatising - http://www.sans.org/y2k/practical/Viriya_Upatising.doc

23/4/43 22:06:48	128.112.80.152	203.146.93.13	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.4	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.7	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.10	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.16	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.21	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.5	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.14	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.28	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.6	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.15	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.29	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.9	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.8	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.11	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.18	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.17	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.23	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.25	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.37	HTTP_TestCgi	URL	/cgi-bin/test-cgi/
23/4/43 22:06:48	128.112.80.152	203.146.93.38	HTTP_TestCgi	URL	/cgi-bin/test-cgi/

Description

This log was taken from the ISS RealSecure HIGH priority event logfile file. We tried to trace back to the host IP address, here is the traceroute result:

```
traceroute to 128.112.80.152 (128.112.80.152), 30 hops max, 40 byte packets
 1  10.15.20.254 (10.15.20.254)  2.391 ms  1.814 ms  1.089 ms
 2  lir6-fl1-1-0.loxinfo.co.th (203.146.43.200)  2.621 ms  1.84 ms  2.968 ms
 3  lir-sp1.loxinfo.net (203.146.64.130)  5.568 ms  2.334 ms  2.955 ms
 4  gip-stock-5-s8-0-6.gip.net (204.59.165.57)  270.716 ms  293.875 ms  297.916 ms
 5  sl-bb10-stk-2-1.sprintlink.net (144.232.4.129)  375.8 ms  390.173 ms  309.367 ms
 6  sl-bb10-ana-6-1.sprintlink.net (144.232.8.90)  224.603 ms  207.489 ms  211.285 ms
 7  pos9-2-155M.lax-bb5.cerf.net (134.24.32.241)  375.717 ms  447.015 ms  494.951 ms
 8  pos4-0-622M.lax-bb4.cerf.net (134.24.32.13)  335.972 ms  362.227 ms  570.589 ms
 9  so1-0-0-622M.dfw-bb2.cerf.net (134.24.29.78)  275.477 ms  372.399 ms  409.056 ms
10  so1-3-0-622M.chi-bb5.cerf.net (134.24.46.82)  346.518 ms  373.824 ms  353.837 ms
11  pos1-0-622M.nyc-bb8.cerf.net (134.24.32.214)  458.897 ms  451.077 ms  441.101 ms
12  pos12-0-0-155M.nyc-bb2.cerf.net (134.24.32.222)  443.448 ms  300.977 ms  498.635 ms
13  princeton-gw.nyc-bb2.cerf.net (134.24.131.6)  445.042 ms  531.366 ms  503.538 ms
14  vgate1.Princeton.EDU (128.112.60.1)  499.049 ms  482.106 ms  641.28 ms
15  *
```

We are probably blocked at the firewall from tracing any further. For vulnerable Web servers, the “test-cgi” security hole allows hacker to arbitrary list remote file in the server.

When we check our TCPDUMP log, we can see a similar behavior as follows:

```
22:06:53.633751 fugue.csbmb.Princeton.EDU.11750 > vweb2.loxinfo.co.th.www: S 1337735200:1337735200(0) win 49152 <mss 1460>
22:06:53.633754 fugue.csbmb.Princeton.EDU.11747 > 203.146.11.129.www: S 1337536800:1337536800(0) win 49152 <mss 1460>
22:06:53.633788 fugue.csbmb.Princeton.EDU.11751 > my.i-kool.com.www: S 1337794400:1337794400(0) win 49152 <mss 1460>
22:06:53.633914 fugue.csbmb.Princeton.EDU.11754 > 203.146.11.136.www: S 1337998400:1337998400(0) win 49152 <mss 1460>
```

```
22:06:53.634242 fugue.csbmb.Princeton.EDU.11755 > 203.146.11.137.www: S 1338056000:1338056000(0) win 49152 <mss 1460>
22:06:53.634447 fugue.csbmb.Princeton.EDU.11764 > 203.146.11.146.www: S 1338625600:1338625600(0) win 49152 <mss 1460>
22:06:53.634569 fugue.csbmb.Princeton.EDU.11765 > 203.146.11.147.www: S 1338701600:1338701600(0) win 49152 <mss 1460>
22:06:53.636965 my.i-kool.com.www > fugue.csbmb.Princeton.EDU.11751: S 2353895647:2353895647(0) ack 1337794401 win 32120 <mss 1460> (DF)
22:06:53.643784 fugue.csbmb.Princeton.EDU.11746 > 203.146.11.128.www: S 1337479200:1337479200(0) win 49152 <mss 1460>
22:06:53.643947 fugue.csbmb.Princeton.EDU.11749 > vweb1.loxinfo.co.th.www: S 1337679200:1337679200(0) win 49152 <mss 1460>
22:06:53.644234 fugue.csbmb.Princeton.EDU.11753 > 203.146.11.135.www: S 1337925600:1337925600(0) win 49152 <mss 1460>
22:06:53.644440 fugue.csbmb.Princeton.EDU.11763 > 203.146.11.145.www: S 1338572800:1338572800(0) win 49152 <mss 1460>
22:06:53.645903 vweb1.loxinfo.co.th.www > fugue.csbmb.Princeton.EDU.11749: R 0:0(0) ack 1337679201 win 0
22:06:53.648986 fugue.csbmb.Princeton.EDU.11748 > vweb0.loxinfo.co.th.www: S 1337607200:1337607200(0) win 49152 <mss 1460>
22:06:53.649149 fugue.csbmb.Princeton.EDU.11752 > 203.146.11.134.www: S 1337868000:1337868000(0) win 49152 <mss 1460>
22:06:53.649356 fugue.csbmb.Princeton.EDU.11762 > 203.146.11.144.www: S 1338498400:1338498400(0) win 49152 <mss 1460>
22:06:53.649643 fugue.csbmb.Princeton.EDU.11766 > 203.146.11.148.www: S 1338762400:1338762400(0) win 49152 <mss 1460>
22:06:53.650481 vweb0.loxinfo.co.th.www > fugue.csbmb.Princeton.EDU.11748: S 2699986351:2699986351(0) ack 1337607201 win 32120 <mss 1460> (DF)
22:06:53.965077 fugue.csbmb.Princeton.EDU.11751 > my.i-kool.com.www: . ack 1 win 2920 (DF)
22:06:53.967492 fugue.csbmb.Princeton.EDU.11750 > vweb2.loxinfo.co.th.www: . ack 3602173860 win 2920 (DF)
22:06:53.968360 fugue.csbmb.Princeton.EDU.11751 > my.i-kool.com.www: P 1:50(49) ack 1 win 32767 (DF)
22:06:53.968863 my.i-kool.com.www > fugue.csbmb.Princeton.EDU.11751: . ack 50 win 32120 (DF)
```

With the TCPDUMP output, you can see pretty clearly that the host is trying to scan Web servers. Once it found the Web server then it started to implement the "test-cgi" attack.

Targeting: Yes
History: No previous history on this host was kept
Techniques: "test-cgi" web attack.
Intent: Unauthorized server access .
Analysis: A program or a script was probably used to probe since the probe speed was quite fast. The intention was quite clear, the hacker was trying to hack your Web server through the "test-cgi" hole.

Severity: 0

Component	Score	Comments
Criticality:	3	no clear target, just a blind sweeping
Lethality:	2	information gathering/confidentiality attack
System Countermeasures:	3	not all our customers will have updated patches
Network Countermeasures:	2	permissive firewall
Severity:	0	(Criticality + Lethality) - (System + Net Countermeasures)

The next trace is typical of something that happens to every class of analysts, they start looking at their traffic, they think they are being scanned and as they look into it, they find, they are the source of the traffic. It is a good thing, a necessary part of the analysis journey we are all on.

I HAVE MET THE ENEMY AND …
Submitted by Paul Stillwell - http://www.sans.org/y2k/practical/Paul_Stillwell.doc

Date: March 12, 2000
Detected using IPChains

At first I thought that this was the tail end of a slow UDP port scan. But the source address and consistent source port (below 1024) were bothering me. The domain is known and the source port was always 123 (xntp). After some research, I discovered that it was actually a misconfiguration of my IPChains rules (I had the source and dest ports reversed) for valid xntp traffic. What through me off initially is that the reverse lookup returns snort.someplace.net. A different name than what is in the xntp configuration file. The config file points to a server called clock.someplace.net which resolves to a.b.c.d, the reverse lookup on a.b.c.d returns snort.someplace.net.

This example illustrates well that caution and objectivity are required when analyzing any detect.

IPChains:

```
<...snip>
Mar 12 04:05:35 hostname kernel: Packet log: input - eth0 PROTO=17 snort.someplace.net:123
me.nowhere.com:61218 L=76 S=0x10 I=63671 F=0x4000 T=245

Mar 12 04:22:37 hostname kernel: Packet log: input - eth0 PROTO=17 snort.someplace.net:123
me.nowhere.com:61219 L=76 S=0x10 I=37095 F=0x4000 T=245

Mar 12 04:39:41 hostname kernel: Packet log: input - eth0 PROTO=17 snort.someplace.net:123
me.nowhere.com:61220 L=76 S=0x10 I=12549 F=0x4000 T=245

Mar 12 04:56:46 hostname kernel: Packet log: input - eth0 PROTO=17 snort.someplace.net:123
me.nowhere.com:61221 L=76 S=0x10 I=54469 F=0x4000 T=245

Mar 12 05:13:50 hostname kernel: Packet log: input - eth0 PROTO=17 snort.someplace.net:123
me.nowhere.com:61222 L=76 S=0x10 I=29903 F=0x4000 T=245
<snip...>
```

This was a hard practical to grade, English is a second language for Javier, but oh my was it clear that he knows what he is doing! For this edition, I am going to do a bit of inline editing to make it easier for you to follow the analysis.

STREAMING MEDIA
Submitted by Javier Romero -

Severity: (3.Criticality + 4.Lethality) – (3.System + 2.Net Countermeasures)=2
Technique: TCP options and timestamp wasting all resources of the service.
Intent: Denial of Service Attack against Microsoft Media Server (Unicast Server).

MS-Streaming Media Server out of service
This server is outside of the firewall on an unprotected network. This server doesn't have network logging capability.

First we see connections to the port 1755. They don't seem to be that harmful, but the streaming media sessions that were running at the time were interrupted.

First Handshake

```
01:57:12.889571 P 10.10.10.1.1082 > x.x.x.A.1755: S 3118948512:3118948512(0) win 16060
<mss 1460,sackOK,timestamp 325246 0,nop,wscale 0> (DF)
                        4500 003c 03e6 4000 4006 c4b1 0a0a 0a01
                        xxxx xxxx 043a 06db b9e7 60a0 0000 0000
                        a002 3ebc 7b06 0000 0204 05b4 0402 080a
                        0004 f67e 0000 0000 0103 0300
01:57:12.889706 P x.x.x.A.1755 > 10.10.10.1.1082: S
977106048:977106048(0) ack 3118948513 win 17520 <mss 1460,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF)
                        4500 0040 053f 4000 8006 8354 xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7480 b9e7 60a1
                        b012 4470 ab00 0000 0204 05b4 0103 0300
                        0101 080a 0000 0000 0000 0000 0101 0402
01:57:12.889897 P 10.10.10.1.1082 > x.x.x.A.1755: . 1:1(0) ack 1 win
16060 <nop,nop,timestamp 325246 0> (DF)
                        4500 0034 03e7 4000 4006 c4b8 0a0a 0a01
                        xxxx xxxx 043a 06db b9e7 60a1 3a3d 7481
                        8010 3ebc fafc 0000 0101 080a 0004 f67e
                        0000 0000
```

First data (we only show 130 bytes)

```
01:57:12.894896 P 10.10.10.1.1082 > x.x.x.A.1755: P 1:177(176) ack 1
win 16060 <nop,nop,timestamp 325247 0> (DF)
                        4500 00e4 03e8 4000 4006 c407 0a0a 0a01
                        xxxx xxxx 043a 06db b9e7 60a1 3a3d 7481
                        8018 3ebc 4f3a 0000 0101 080a 0004 f67f
                        0000 0000 0100 0000 cefa 0bb0 a000 0000
01:57:12.963644 P x.x.x.A.1755 > 10.10.10.1.1082: P 1:257(256) ack 177 win 17344
<nop,nop,timestamp 4869838 325247> (DF)
                        4500 0134 0540 4000 8006 825f xxxx xxxx
                        c80e f105 06db 043a 3a3d 7481 b9e7 6151
                        8018 43c0 a753 0000 0101 080a 004a 4ece
                        0004 f67f 0100 0000 cefa 0bb0 f000 0000
01:57:12.964035 P 10.10.10.1.1082 > x.x.x.A.1755: P 177:713(536) ack 257 win 15804
<nop,nop,timestamp 325254 4869838> (DF)
                        4500 024c 03e9 4000 4006 c29e 0a0a 0a01
                        xxxx xxxx 043a 06db b9e7 6151 3a3d 7581
                        8018 3dbc 2110 0000 0101 080a 0004 f686
```

```
                        004a 4ece 0100 0000 cefa 0bb0 2000 0000
01:57:12.964342 P x.x.x.A.1755 > 10.10.10.1.1082: P 257:769(512) ack 713 win 16808
<nop,nop,timestamp 4869838 325254> (DF)
                        4500 0234 0541 4000 8006 815e xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7581 b9e7 6369
                        8018 41a8 5c0b 0000 0101 080a 004a 4ece
                        0004 f686 0100 0000 cefa 0bb0 f001 0000
01:57:12.964433 P x.x.x.A.1755 > 10.10.10.1.1082: P 769:1281(512) ack 713 win 16808
<nop,nop,timestamp 4869838 325254> (DF)
                        4500 0234 0542 4000 8006 815d xxxx xxxx
                        c80e f105 06db 043a 3a3d 7781 b9e7 6369
                        8018 41a8 414e 0000 0101 080a 004a 4ece
                        0004 f686 0100 0000 cefa 0bb0 f001 0000
01:57:12.964614 P x.x.x.A.1755 > 10.10.10.1.1082: P 1281:2337(1056) ack
713 win 16808 <nop,nop,timestamp 4869838 325254> (DF)
                        4500 0454 0543 4000 8006 7f3c xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7981 b9e7 6369
                        8018 41a8 abe8 0000 0101 080a 004a 4ece
                        0004 f686 0100 0000 cefa 0bb0 1004 0000
01:57:12.964831 P x.x.x.A.1755 > 10.10.10.1.1082: P 2337:2417(80) ack 713 win 16808
<nop,nop,timestamp 4869838 325254> (DF)
                        4500 0084 0544 4000 8006 830b xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7da1 b9e7 6369
                        8018 41a8 40b1 0000 0101 080a 004a 4ece
                        0004 f686 0100 0000 cefa 0bb0 4000 0000
01:57:12.965541 P x.x.x.A.1755 > 10.10.10.1.1082: P 2417:2569(152) ack
713 win 16808 <nop,nop,timestamp 4869838 325254> (DF)
                        4500 00cc 0545 4000 8006 82c2 xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7df1 b9e7 6369
                        8018 41a8 403f 0000 0101 080a 004a 4ece
                        0004 f686 0100 0000 cefa 0bb0 8800 0000
01:57:12.965695 P x.x.x.A.1755 > 10.10.10.1.1082: P 2569:2625(56) ack 713 win 16808
<nop,nop,timestamp 4869838 325254> (DF)
                        4500 006c 0546 4000 8006 8321 xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7e89 b9e7 6369
                        8018 41a8 7447 0000 0101 080a 004a 4ece
01:57:12.965853 P x.x.x.A.1755 > 10.10.10.1.1082: P 2625:2673(48) ack
713 win 16808 <nop,nop,timestamp 4869838 325254> (DF)
                        4500 0064 0547 4000 8006 8328 xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7ec1 b9e7 6369
                        8018 41a8 cb17 0000 0101 080a 004a 4ece
                        0004 f686 0100 0000 cefa 0bb0 2000 0000
01:57:12.971943 P 10.10.10.1.1082 > x.x.x.A.1755: . 713:713(0) ack 2673
win 13388 <nop,nop,timestamp 325255 4869838> (DF)
                        4500 0034 03ea 4000 4006 c4b5 c80e f105
                        xxxx xxxx 043a 06db b9e7 6369 3a3d 7ef1
                        8010 344c a913 0000 0101 080a 0004 f687
01:57:24.102639 P x.x.x.A.1755 > 10.10.10.1.1082: R
977108721:977108721(0) win 0 (DF)
                        4500 0028 0549 4000 8006 8362 xxxx xxxx
                        0a0a 0a01 06db 043a 3a3d 7ef1 b9e7 6369
                        5004 0000 5c27 0000 0770 7561 7169
```

Analysis

1. This transaction hides the real intention. In general we can say that the TCP Options are using in several transactions with this particular service for example. Initially it is a little difficult to determine why the service is stopped. The session is alive and continuous transactions are still occuring, but the video casting goes down.

2. If you observe the 12th octet of TCP header, you can see that TCP OPTIONS is active. Due to a vulnerability in the MS Streaming Media Server, the option called TIMESTAMP allows the attacker to consume the resources from the system. The signature is underlined in red.

3. We can see continuous arrival of packets with timestamp option. By reviewing the Microsoft web page, we can find a patch to stop this attack.

4. This attack will evade many IDS systems because this is a valid transaction.

5. One thing to note about this detect, the three-way TCP handshake must complete or the DoS attack will never occur.

6. Your security team should apply the correct patch against this Denial-of-Service attack. Also if you prepare a correct filter to equip your IDS for this kind of DoS attack, your team will be able to catch the intruder, for example:

   tcpdump –w –i eth1 /home/username/msms-dos-filter.in
   ´(tcp and ((tcp[12] & 0xf0 >=112) and (tcp[13] & 0x1a !=0)) and port 1755)´

   Where:
   (tcp[12] & 0xf0 >=112) Is used for packets with TCP options greater than 2 bytes

   112 dec. = 70 hex (7 is the number of bytes in TCP header)
   (tcp[13] & 0x1a !=0)

   Is used for packets with SYN, PSH, ACK.
7. Remember that DoS are difficult to catch, but you can reduce the time to capture the intruder with a good filter.

It still isn't clear to me why we see so much probing for linuxconf on the network, but we certainly do. In addition to the risk the analyst proposes in this trace, RedHat 5.1 had a denial of service that could be executed and information about a buffer overflow has been posted to the net.

LINUXCONF
Submitted by Jay Howie - http://www.sans.org/y2k/practical/Jay_Howie.doc

Apr 19 22:37:06 cc1014244-a kernel: securityalert: tcp if=ef0 from 24.1.86.154:1198 to 24.3.21.199 on unserved port 98

Evidence of Active Targeting: Yes

History: No history from the GIAC web site reported.

Existence: Performed nslookup on the IP address 24.1.86.154 and it was resolved to an @Home user c685583-a.pinol1.sfba.home.com

Technique: This is a scan targeting the TCP port 98 which is known to be used by the linuxconf administration utility.

Evidence of Intent: To remotely administer the system via the linuxconf administration tool.

Analysis:

- Trace shows attacker looking for TCP port 98 (linuxconf) response.
- Linuxconf is an administration system (GUI) for the Linux operating system Linuxconf runs as root therefore if compromised, the attacker could potentially gain root access.

Severity:
(criticality + lethality) – (system countermeasure + network countermeasure)
(5 + 5) – (4 + 4) = 2

Criticality:	Targeted machine.
Lethality:	Attempted connection to a malicious Trojan.
System Countermeasure:	Make sure linuxconf, has been configured correctly.
Network Countermeasure:	Proper configuration and deployment of firewall technology and IDS may help prevent this from happening or at least being unknown.

Here we have another great example of running a trace to ground. It is certainly worth being familiar with the behavior of Jetdirect, since HP printers are so popular, this type Jetdirect created activity will show up fairly often.

JET DIRECT PRINTER IP MIX UP
Submitted by Mike Harvey - http://www.sans.org/y2k/practical/Mike_Harvey.txt

Reference the tcpdump output at the end of this analysis, labeled TCPDUMP TRACE.

GENERAL INFORMATION

This is a sanitized version of an actual network trace from a host on my network. The trace is taken from a host that I administer within our network.

What made this trace stand out is that regular and frequent connection attempts were comming from my host and directed to a remote host on my network. The remote host has no association with my host. Thoughts of trojans filled my mind...

ANALYSIS:

- About every 5 - 10 seconds my host (myhost) sends a tcp connection attempt from a varying high-numbered port to the remote host (remotehost) port 9100. The number of connection attempts using the same source port varies from one to a handful, then the source port changes.
- The remote host does not respond.
- Looking up port numbers on the internet, I find that port 9100 is the standard port used by networked laserjet printers. Searching the network traces shows other connections to destination port 9100 to machines that are definitely networked laserjet printers.
- Looking in myhost's print queue, a print job has been trying to print to a remote laserjet printer for about a month. The IP address of this printer in the hosts file is correct and does not match the IP for remotehost. However, looking back in the system change log, we find that the current IP of remotehost is the same as the original IP of the printer the print queue is going to. This IP was updated in /etc/hosts when the network was subnetted and new IP's assigned.
- Printing on myhost is done via HP's jetdirect software.
- When the print job was cancelled the tcp connection attemps to remotehost stopped.
- When the printer was removed and re-added with the correct IP address via jetdirect, printing went to the printer and not to remotehost.

EVIDENCE OF TARGETING

Myhost is definitely targeting remotehost with these tcp connection attempts.

THEORIES OF INTENT

- MY UNIX HOST THINKS THIS POOR PC IS AN HP PRINTER: There is no doubt that this is the case. The problem was that when the printer's IP was changed, it was updated in /etc/hosts, but no changes were made in the jetdirect printer software. Somehow jetdirect or the UNIX print service remembers IP addresses and does not look in /etc/hosts to resolve printers. Remotehost just happened to be assigned the old IP of the printer. Removing and re-adding the printer in jetdirect fixed the problem. Nothing malicious going on here.

----- TCPDUMP TRACE -----

```
08:27:36.603367 myhost.mynet.com.63766 > remotehost.mynet.com.9100: S
2784969512:2784969512(0) win 65535 <nop,wscale 1,mss 1460>
08:27:40.102674 myhost.mynet.com.63766 > remotehost.mynet.com.9100: S
2784969512:2784969512(0) win 65535 <nop,wscale 1,mss 1460>
08:27:46.502821 myhost.mynet.com.63766 > remotehost.mynet.com.9100: S
2784969512:2784969512(0) win 65535 <nop,wscale 1,mss 1460>
08:27:59.303052 myhost.mynet.com.63766 > remotehost.mynet.com.9100: S
2784969512:2784969512(0) win 65535 <nop,wscale 1,mss 1460>
08:28:24.903372 myhost.mynet.com.63766 > remotehost.mynet.com.9100: S
2784969512:2784969512(0) win 65535 <nop,wscale 1,mss 1460>
08:29:16.104243 myhost.mynet.com.63766 > remotehost.mynet.com.9100: S
2784969512:2784969512(0) win 65535 <nop,wscale 1,mss 1460>
08:31:58.517446 myhost.mynet.com.63794 > remotehost.mynet.com.9100: S
2819183594:2819183594(0) win 65535 <nop,wscale 1,mss 1460>
08:32:02.016897 myhost.mynet.com.63794 > remotehost.mynet.com.9100: S
2819183594:2819183594(0) win 65535 <nop,wscale 1,mss 1460>
08:32:12.027648 myhost.mynet.com.63795 > remotehost.mynet.com.9100: S
2820921324:2820921324(0) win 65535 <nop,wscale 1,mss 1460>
08:32:22.037771 myhost.mynet.com.63796 > remotehost.mynet.com.9100: S
2822302968:2822302968(0) win 65535 <nop,wscale 1,mss 1460>
08:32:32.048019 myhost.mynet.com.63797 > remotehost.mynet.com.9100: S
2823664783:2823664783(0) win 65535 <nop,wscale 1,mss 1460>
08:32:42.058064 myhost.mynet.com.63798 > remotehost.mynet.com.9100: S
2824968122:2824968122(0) win 65535 <nop,wscale 1,mss 1460>
08:32:52.068293 myhost.mynet.com.63799 > remotehost.mynet.com.9100: S
2826275463:2826275463(0) win 65535 <nop,wscale 1,mss 1460>
```

Over the years we have had our intrusion detection systems alert on any number of vulnerability assessment tool scans. This always makes the security shop look good when they catch these, but bad when they don't since these tools are not whatsoever subtle. Here we have a detect of one component of a scan.

GET FILE
Submitted by Stephen Zvacek - http://www.sans.org/y2k/practical/Stephen_Zvacek.doc

Detect #2, Get file

```
 1 M      [xxx.210.230.198] [xxx.89.130.193]     60 0:00:00.000   0.000.000     04/06/2000 10:05:31 AM TCP: I
S=1198 SYN SEQ=508304410 LEN=0 WIN=8192
     2        [xxx.89.130.193]  [xxx.210.230.198]    60 0:00:00.001   0.001.062     04/06/2000 10:05:31 AM TC
D=1198 S=80 SYN ACK=508304411 SEQ=3210260865 LEN=0 WIN=8760
     3        [xxx.210.230.198] [xxx.89.130.193]     60 0:00:00.063   0.062.786     04/06/2000 10:05:31 AM TC
S=1198     ACK=3210260866 WIN=8760
     4        [xxx.210.230.198] [xxx.89.130.193]     83 0:00:00.065   0.001.531     04/06/2000 10:05:31 AM HT
Port=1198 GET /perl/files.pl HTTP/1.0
     5        [xxx.89.130.193]  [xxx.210.230.198]    60 0:00:00.066   0.001.198     04/06/2000 10:05:31 AM TC
D=1198 S=80     ACK=508304440 WIN=8760
     6        [xxx.210.230.198] [xxx.89.130.193]     60 0:00:00.123   0.056.584     04/06/2000 10:05:31 AM HT
Port=1198 HTML Data
     7        [xxx.89.130.193]  [xxx.210.230.198]   215 0:00:00.137   0.013.866     04/06/2000 10:05:31 AM HT
Port=1198 HTML Data
     8        [xxx.89.130.193]  [xxx.210.230.198]   261 0:00:00.137   0.000.498     04/06/2000 10:05:31 AM HT
Port=1198 HTML Data
     9        [xxx.89.130.193]  [xxx.210.230.198]    60 0:00:00.137   0.000.385     04/06/2000 10:05:31 AM TC
D=1198 S=80 FIN ACK=508304442 SEQ=3210261234 LEN=0 WIN=8760
    10        [xxx.210.230.198] [xxx.89.130.193]     60 0:00:00.191   0.053.966     04/06/2000 10:05:31 AM TC
S=1198 FIN ACK=3210261027 SEQ=508304442 LEN=0 WIN=8599
    11        [xxx.89.130.193]  [xxx.210.230.198]    60 0:00:00.192   0.000.705     04/06/2000 10:05:31 AM TC
D=1198 S=80     ACK=508304443 WIN=8760
    12        [xxx.210.230.198] [xxx.89.130.193]     60 0:00:00.195   0.003.246     04/06/2000 10:05:31 AM TC
S=1198 RST WIN=0
    13        [xxx.210.230.198] [xxx.89.130.193]     60 0:00:00.196   0.000.802     04/06/2000 10:05:31 AM TC
S=1198 RST WIN=0
    14        [xxx.210.230.198] [xxx.89.130.193]     60 0:00:00.255   0.059.328     04/06/2000 10:05:31 AM TC
S=1198 RST WIN=0
```

ANALYSIS

Evidence` of Active Targeting?

Yes. This appeared on our external Internet segment. This trace is one in a series of GET/POST actions initiated by this remote m

Identify the Technique?

The remote machine has made a TCP connection with our webserver. It is attempting to retrieve (GET) the file "files.pl"from our webserver. Although the trace printout does lists packets 6 through 16 as HTML data, the server responded the file was not found problem in the 'files.pl' script distributed with the Novell WebServer Examples Toolkit v2 could allow a remote attacker to view th of any file or directory on vulnerable servers. The attacker would be limited to viewing files accessible to the user owning the serv (This is from the ISS Security Scanner database on specific checks made by this program)

Evidence of Intent?

There is clearly intent from this trace.

Identify Hostile Individuals and Groups

It was later identified that we were being evaluated by our government contracting office. They tasked another contrac evaluate the security of our external connections. It appeared from this and other traces they were using ISS Security to perform the task.

Severity?

(2+2)-(3+4) = -3
Criticality - This machine is a webserver
Lethality - This could be a confidentiality issue
Countermeasures - Older Unix system with some patches missing
Net Countermeasures - Good firewall with some external connections