David Thibault
11/17/2000
PRACTICAL ASSIGNMENT FOR
GIAC GCIA CERTIFICATION
Network Security 2000

TABLE OF CONTENTS

PREFACE

DESCRIPTION OF TRACE SOURCES:

HOME OFFICE:

These traces were taken from the gateway box on my home office network. I have one real IP address, and a protected LAN behind it using unroutable addresses. The gateway is a Linux box running RedHat 6.0, doing IP-Masquerading. This box is connected via VPN to two other offices of the company I work for. From here I have access to the entire network. Therefore, although there is not much inside my protected LAN, from this box the entire company network is vulnerable. It is obvious, then, that the protection of this box is critical. It is connected to the internet via a cable modem, so this only increases the level of alertness that must be kept. I am currently running Snort 1.63 Patch 2, the Shadow IDS, IPChains and the Portsentry by Psionic Software. Snort has its own alert log format, Shadow output is in tcpdump format, and Portsentry and IPChains log to the Linux SYSLOG. This gateway box will be referred to as **mylinuxgw.homeoffice.net** and **11.22.33.44** for it's real IP address, for the protected

LAN I will use **10.0.0.0/24**. Also, my own address will be bolded wherever it occurs.

COMPANY FOREIGN OFFICE:
These traces were taken from the Cisco router logs in a foreign office of the company I work for. There are a very small number of workstations and one server there. We are in the process of scheduling a sniffer install there, but as yet there is none.

LOG FILE FORMAT:

Linux SYSLOG (format used by IPChains):

Date Time Hostname Message_Source: Message_Type: Chain_name ACTION interface PROTOCOL srcaddr:port destaddr:port Length [SYN_FLAG] packet_id [FIN_FLAG] Time-to-live

Example (NOTE—All traces I list will be in italics):

```
Nov 19 15:38:34 mylinuxgw kernel: Packet log: input DENY eth0 PROTO=6
63.194.16.45:1609 11.22.33.44:21 L=60 S=0x00 I=31699 F=0x4000 T=50
```

Cisco SYSLOG format:

Date Time routername : 14w5d: Access Log Name: ACL number action protocol source_address(port) -> destination_address-> port

Example:

```
Nov 19 21:07:49 6X:router.officea.net 5372: 14w5d: %SEC-6-IPACCESSLOGP:
list 106 permitted udp wkstn2.officea.net(137) -> 190.10.92.2(137)
```

Linux SYSLOG (format used by Portsentry):

Date Time Hostname portsentry: warning_type: Scan Type (with flags): src_addr_name/src_addr_num to dest protocol port:XXX

Example:

```
Nov 18 04:52:42 mylinuxgw portsentry[868]: attackalert: Unknown Type: Packet
Flags: SYN: 0 FIN: 0 ACK: 0 PSH: 1 URG: 0 RST: 0 from host:
210.111.4.168/210.111.4.168 to TCP port: 23
```

Tcpdump log format (as used by Shadow IDS):

Timestamp Source.port Dest.port flags Beginning_seq_# : Ending_Seq_#(bytes) options

Example:

```
08:14:47.158382 cc644109-a.hwrd1.md.home.com.sunrpc
>mylinuxgw.homeoffice.net.sunrpc: SF 2002853141:2002853141(0) win 1028
```

Snort alert format:

[**]ALERT NAME[**]
Date-Time srcaddr:port -> destaddr:port
PROTOCOL Time-to-live Type_of_service packet_id
FLAGS Seq_num: 0x00000 Ack_num: 0x0000 windowsize: 0x000

Example:

```
[**] SCAN-SYN FIN [**]
11/16-08:14:47.158382 24.3.24.41:111 -> 11.22.33.44:111
TCP TTL:30 TOS:0x0 ID:39426
******SF Seq: 0x77611D15   Ack: 0x121B478   Win: 0x404
```

Snort log format:

All Snort processes have been logging to tcpdump binary format, so the format will be the same as tcpdump output.

NETWORK DETECTS

DETECT NUMBER 1: TCP STEALTH SCANS

SYN-FIN Scan (Source—HOME OFFICE):

Tcpdump for this detect (format):

```
08:14:47.158382 cc644109-a.hwrd1.md.home.com.sunrpc >mylinuxgw.homeoffice.net.sunrpc: SF 2002853141:2002853141(0) win 1028
```

Associated SYSLOG info (IPChains format):

```
Nov 16 08:14:47 mylinuxgw kernel: Packet log: input DENY eth0 PROTO=6 24.3.24.41:111 11.22.33.44:111 L=40 S=0x00 I=39426 F=0x0000 T=30
```

Associated Snort alert (format):

```
[**] SCAN-SYN FIN [**]
11/16-08:14:47.158382 24.3.24.41:111 -> 11.22.33.44:111
TCP TTL:30 TOS:0x0 ID:39426
******SF Seq: 0x77611D15   Ack: 0x121B478   Win: 0x404
```

SPOOFED ?…

The detect above was a solitary packet that came in from the outside interface on my Linux gateway. Given the fact that both the SYN and FIN flags were set, it was clearly crafted. It did not come in with a flurry of other crafted packets. Also notable is the fact that the sender went straight for the portmapper. My first instinct would therefore be that this is the type of behavior where it would be common to NOT spoof one's source address. If there were a flurry of packets coming in from a bunch of different source addresses, I would guess that only one was not spoofed and the rest were merely decoys. However, recent traces in the security community have come up with a second alternative, known as a "spoof bounced" port scan. See the mechanism of the attack below…

DESCRIPTION OF ATTACK…

This type of attack, as mentioned above, could be one of two things. To begin with, sending a single packet to a critical port like portmapper tells me that the attacker could very well be sweeping entire subnets looking for UNIX boxes with unprotected portmapping services. By sending only one packet, the sender probably hoped to map my machine making as little noise as possible. The second possibility would be the "Spoof Bounced" portscan attack described in depth in the Attack Mechanism section below. This latter possibility certainly carries with it more "danger and intrigue", since it would require a lot more creativity on the part of the attacker. However, intrigue may not be the case either, since the Correlations section of the paper discusses a theory of another SANS member who has come up with some interesting ideas that would put this into the hands of the script kiddies.

Also, the associated SYSLOG entry tells me that my firewall stopped this packet in its tracks. It is clear that the IPChains INPUT chain applied itself here in order to block this packet from actually hitting my portmapper. I have disabled ICMP Port & Host Unreachable messages in the OUTPUT chain, so the attacker didn't get any response whatsoever from my box. I have not received any other traffic from this IP address. Since this was just a reconnaissance mission, the purpose was simply to get some sort of response. Fortunately, the attacker got nothing. If I did give up a response here, the attacker could have proceeded to one of many options based on portmapper vulnerabilities. However, the next logical step would have been OS fingerprinting in an effort to discern which exact version of operating system and portmapper I was running, so as to maximize the effect of an attempted portmapper exploit.

ATTACK MECHANISM…
This attack is simply meant to collect information. The more unlikely (or at least less intelligent) attack here is that the attacker sent a single SYN-FIN packet to every machine on a subnet to TCP port 111 without spoofing his/her address. I say this is less intelligent only because this particular packet is an impossible combination of flags, and will no doubt send Snort into a frenzy. In the case of my Linux box, it will respond differently if there is something running on that port or not. If you send a SYN-FIN to an open port on a Linux then you will get back a SYN-ACK. If you send it to a closed port, you get a RESET-ACK. Therefore, if this packet got through the firewall, I would have divulged a lot about my gateway box just by sending back a single response packet. Fortunately, my firewall stopped both the incoming packet from entering (so therefore no TCP responses from leaving) and any ICMP unreachable responses from leaving.

The more difficult of the two attacks discussed in the Description section above is as follows. The are 3 hosts involved: the attacking host, a middle-man, and a target host. The attacking host starts TCP pinging the middle-man, which is a box that HAS to fit a certain profile. The middle-man host must be a "quiet host" on the internet (meaning it is currently experiencing no internet traffic). By TCP pinging this host we see that the host is spitting out IP ID's that are incrementing by 1 CONSISTENTLY. Here's a sample, using hping2 to send a SYN packet to a quiet WinNT box (using the command "hping –W unsuspecting.middleman.com"):

```
46 bytes from middle.man.com: flags=RA seq=0 ttl=128 id=1213 win=0 rtt=0.4 ms
46 bytes from middle.man.com: flags=RA seq=1 ttl=128 id=1214 win=0 rtt=0.3 ms
46 bytes from middle.man.com: flags=RA seq=2 ttl=128 id=1215 win=0 rtt=0.3 ms
46 bytes from middle.man.com: flags=RA seq=3 ttl=128 id=1216 win=0 rtt=0.3 ms
46 bytes from middle.man.com: flags=RA seq=4 ttl=128 id=1217 win=0 rtt=0.2 ms
46 bytes from middle.man.com: flags=RA seq=5 ttl=128 id=1218 win=0 rtt=0.2 ms
```

Note the IP ID's incrementing each time. Then while this ping is going, in a different terminal window the attacker sends a set of TCP pings to the target host with the source address spoofed as the middle man. Also, if we are to match this attack pattern, we could set the IP ID to 39426, and set the source and destination ports to the same port. In this example we'll use tcp/135 (an open port on an NT box) and tcp/23 (telnet—a closed port on an NT box). Here are the results:

(note: This is a recreation of the attack on a protected network. This is not a trace collected "in the wild")

OPEN PORTS:

TERMINAL WINDOW
1: TERMINAL WINDOW 2:

[root@attacker /root]# **hping middle.man.com -Wr** eth0 default routing interface selected (according to /proc) HPING middle.man.com (eth0 middle.man.com): NO FLAGS are set, 40 headers + 0 data bytes *46 bytes from middle.man.com: flags=RA seq=0 ttl=128 id=6406 win=0 rtt=0.5 ms*	[root@attacker / **135 -N 39426** eth0 default rout HPING target.b

46 bytes from middle.man.com: flags=RA seq=1 ttl=128 id=+1 win=0 rtt=0.4 ms ***46 bytes from middle.man.com: flags=RA seq=2 ttl=128 id=+2 win=0 rtt=0.4 ms*** ***46 bytes from middle.man.com: flags=RA seq=3 ttl=128 id=+2 win=0 rtt=0.3 ms*** *46 bytes from middle.man.com: flags=RA seq=4 ttl=128 id=+1 win=0 rtt=0.4 ms*	

I've bolded the tcp ping above during the part that we see the middle-man skipping IP ID's. Below is a Tcpdump trace of the whole attack:
(NOTE: Hping (above in the table) has a switch for interpreting Windows IP ID's. With the –W, it produces the incrementing IP ID's above. However, it is clear from the trace below that tcpdump mangles them)

```
18:34:33.669535 attacker.badguy.net.1215 > middle.man.com.0: . win 512 (ttl 64, id 1787)
18:34:33.669741 middle.man.com.0 > attacker.badguy.net.1215: R 0:0(0) ack 1575966427 win 0 (ttl 128, id 1561)
18:34:34.660155 attacker.badguy.net.1216 > middle.man.com.0: . win 512 (ttl 64, id 33140)
18:34:34.660330 middle.man.com.0 > attacker.badguy.net.1216: R 0:0(0) ack 961495062 win 0 (ttl 128, id 1817)
18:34:35.015536 middle.man.com.135 > target.box.com.135: SF 1763821839:1763821839(0) win 512 (ttl 64, id 39426)
18:34:35.015965 target.box.com.135 > middle.man.com.135: S 18827712:18827712(0) ack 1763821840 win 8576 <mss 1460> (DF) (ttl 128, id 56975)
18:34:35.016097 middle.man.com.135 > target.box.com.135: R 1763821840:1763821840(0) win 0 (ttl 128, id 2073)
18:34:35.660098 attacker.badguy.net.1217 > middle.man.com.0: . win 512 (ttl 64, id 7329)
18:34:35.660255 middle.man.com.0 > attacker.badguy.net.1217: R 0:0(0) ack 1413144586 win 0 (ttl 128, id 2329)
18:34:36.010087 middle.man.com.136 > target.box.com.135: SF 895651928:895651928(0) win 512 (ttl 64, id 39426)
18:34:36.010303 target.box.com.135 > middle.man.com.136: S 18828704:18828704(0) ack 895651929 win 8576 <mss 1460> (DF) (ttl 128, id 57231)
18:34:36.010425 middle.man.com.136 > target.box.com.135: R 895651929:895651929(0) win 0 (ttl 128, id 2585)
18:34:36.660100 attacker.badguy.net.1218 > middle.man.com.0: . win 512 (ttl 64, id 36139)
18:34:36.660255 middle.man.com.0 > attacker.badguy.net.1218: R 0:0(0) ack 380218303 win 0 (ttl 128, id 2841)
18:34:37.660103 attacker.badguy.net.1219 > middle.man.com.0: . win 512 (ttl 64, id 54530)
18:34:37.660268 middle.man.com.0 > attacker.badguy.net.1219: R 0:0(0) ack 2098752387 win 0 (ttl 128, id 3097)
```

The lines highlighted in red above are the only traffic that the target box sees. The black lines are the attacker tcp pinging the middle-man to watch the IP ID's (as seen in the left-hand box of the table above). Note that nowhere in those red lines is the address of the attacker. When sending a SYN-FIN to an open port on the Windows box, then, the target box replies with a SYN-ACK to the middle-man whose IP we spoofed in the SYN-FIN. The middle-man replies "I didn't ask for that!?!?!" with a RESET. This RESET sent back to the target eats up 1 IP ID, which is why we see the two lines in the left of the table above (labeled TERMINAL 1) skipping IP ID's twice (and two sets of red lines in the Tcpdump trace above). Now for a closed port:

CLOSED PORTS:

TERMINAL WINDOW 1:	TERMINAL WINDOW 2:
[root@attacker /root]# **hping middle.man.com -Wr** eth0 default routing interface selected (according to /proc)	[root@attacker /roc **N 39426**

HPING middle.man.com (eth0 middle.man.com): NO FLAGS are set, 40 headers + 0 data bytes *46 bytes from middle.man.com: flags=RA seq=0 ttl=128 id=6454 win=0 rtt=0.5 ms* *46 bytes from middle.man.com: flags=RA seq=1 ttl=128 id=+1 win=0 rtt=0.3 ms* *46 bytes from middle.man.com: flags=RA seq=2 ttl=128 id=+1 win=0 rtt=0.4 ms* *46 bytes from middle.man.com: flags=RA seq=3 ttl=128 id=+1 win=0 rtt=0.4 ms* *46 bytes from middle.man.com: flags=RA seq=4 ttl=128 id=+1 win=0 rtt=0.3 ms* *46 bytes from middle.man.com: flags=RA seq=5 ttl=128 id=+1 win=0 rtt=0.8 ms*	eth0 default routing HPING target.box

Here we see no skipping IP ID's. Let's look at the Tcpdump trace to find out why:

```
18:37:22.777321 attacker.badguy.net.2087 > middle.man.com.0: . win 512 (ttl 64, id 50831)
18:37:22.777515 middle.man.com.0 > attacker.badguy.net.2087: R 0:0(0) ack 1161313143 win 0
(ttl 128, id 13849)
18:37:23.770090 attacker.badguy.net.2088 > middle.man.com.0: . win 512 (ttl 64, id 30774)
18:37:23.770250 middle.man.com.0 > attacker.badguy.net.2088: R 0:0(0) ack 633758490 win 0
(ttl 128, id 14105)
18:37:24.770146 attacker.badguy.net.2089 > middle.man.com.0: . win 512 (ttl 64, id 32322)
18:37:24.770322 middle.man.com.0 > attacker.badguy.net.2089: R 0:0(0) ack 1101532844 win 0
(ttl 128, id 14361)
18:37:24.949038 middle.man.com.23 > target.box.com.23: SF 1910664179:1910664179(0) win
512 (ttl 64, id 39426)
18:37:24.949247 target.box.com.23 > middle.man.com.23: R 0:0(0) ack 1910664181 win 0 (ttl
128, id 63887)
18:37:25.770098 attacker.badguy.net.2090 > middle.man.com.0: . win 512 (ttl 64, id 58936)
18:37:25.770263 middle.man.com.0 > attacker.badguy.net.2090: R 0:0(0) ack 584670924 win 0
(ttl 128, id 14617)
18:37:25.940137 middle.man.com.24 > target.box.com.23: SF 144655996:144655996(0) win 512
(ttl 64, id 39426)
18:37:25.940327 target.box.com.23 > middle.man.com.24: R 0:0(0) ack 144655998 win 0 (ttl
128, id 64143)
18:37:26.770096 attacker.badguy.net.2091 > middle.man.com.0: . win 512 (ttl 64, id 1186)
18:37:26.770253 middle.man.com.0 > attacker.badguy.net.2091: R 0:0(0) ack 476932899 win 0
(ttl 128, id 14873)
18:37:27.770706 attacker.badguy.net.2092 > middle.man.com.0: . win 512 (ttl 64, id 52441)
18:37:27.771014 middle.man.com.0 > attacker.badguy.net.2092: R 0:0(0) ack 1475668273 win 0
(ttl 128, id 15129)
```

Once again, I've highlighted traffic showing up on the target network in red, and the traffic between the attacker and the middle-man stays black. As we see above, the SYN-FIN sent to a closed port (telnet port, Windows box used, no telnet…) the target sends back a RESET immediately. Therefore, no more responses are made from the middle-man to the target, so no extra IP ID's are used. Therefore the IP ID's never vary, always incrementing by 1 on the middle-man, as seen in the CLOSED PORTS, TERMINAL WINDOW 1 above. Therefore, the attacker now knows that this port is closed.

CORRELATIONS…

The section below illustrates another example of this type of scan. The excerpt below was taken from a post by Terry Bidwell on the SANS website at http://www.sans.org/y2k/111600.htm. In this article, Terry suggests that this type of scan is part of a complicated 3-way spoofed source bounced network scan.

> "There have been several postings to security mailing lists asking about IDS network

> detects showing SYN-FIN port cans where the source port == the destination port, the IP ID is always 39426, and the window size is always 1028 (0x404).
>
> ```
> [**] SCAN-SYN FIN [**]
> 10/26-18:14:45.311283 209.1.2.102:21 -> my.host1.com:21 TCP TTL:30
> TOS:0x0 ID:39426 **SF**** Seq: 0x117EB2A6 Ack: 0x2CB3E404 Win: 0x404”
> ```

I created the examples in the Attack Mechanism section above using Hping2. The packets generated by the examples above match the signature in the detect I had above and the detect Terry Bidwell cites. However, the fact that there are many detects popping up with this same signature suggests, as Terry points out, that someone has a modified version of Idlescan (or some other tool) floating around.

EVIDENCE OF ACTIVE TARGETING?…

This attack by definition takes at least some active targeting. It requires that someone actively target a quiet host on the internet in order to use it as a middle-man. This attack won’t work with a busy host. The end target could be scripted in batches, though. As for the port they were going after on my box, that is definitely active targeting. First of all, finding something responding to TCP pings on port 111 would indicate a UNIX box, so that is the beginning of an OS fingerprint. However, since this is the only packet I received from this IP address, I would say that one of two things has occurred. Either the attacker did script a batch of IP addresses as target hosts using the same middle-man, effectively sweeping entire subnets, or the attacker has more than one compromised machine from which to attack me. I would say that since I have not seen any other TCP pings with matching source and destination ports hitting my box, the attacker is at least trying different attack angles from those multiple boxes.

SEVERITY…

A common measure of the severity of an attack is represented by the following equation:

Severity = (Criticality of Target + Lethality of the Attack) - (Host-based Countermeasures + Network Countermeasures)

Assigning a scale of 1-5 (1 being lowest and 5 being highest) to each of these variables gives us a relative number representing the severity of the attack. In the case of the detect that hit my machine, here’s how this breaks down:

Criticality of Target: This machine acts as the router for my LAN, as well as the IPSec VPN to the other two offices in my company. Therefore a break-in here would compromise more than just my LAN. In that respect, this gets a ***5***.

Lethality of Attack: This attack failed, because the packet was stopped by the IPChains input ruleset, and the output ruleset stops ICMP host & port unreachables. Therefore I’d give this variable a ***1***.

Host-based Countermeasures: This host had Portsentry by Psionic running, so if the packet had gotten through and the attacker had hit more than 2 ports, he would have gotten his middle-man IP address locked out by IPChains & Tcp-Wrappers. In this case, though, he would have successfully mapped at least 2 ports. Therefore, I’d give this host-based defense a ***4***.

Network Countermeasures: This host had IPChains configured to stop this packet, so it didn't even get through to my TCP stack. Also, it had Snort, which screamed about the impossible flags on this packet. For these reasons I'd give this variable a ***5***.
***Severity = (Criticality + Lethality) – (Host Countermeasures + Net Countermeasures)* = (5 + 1) – (4 + 5) = -3**

DEFENSIVE RECOMMENDATION…
It is evidenced that in this case a well configured firewall and IDS was invaluable in both detecting this attempted hack and preventing its success. Beyond this, a stateful inspection or application gateway firewall would be an even better recommendation. In this case I had Portsentry in Advanced Stealth Scan mode. In this mode, when it starts up it checks which ports are listening for valid traffic and listens to *all other* ports. Therefore, if I'm not running telnetd, and someone tries to telnet to me, that's one strike against them. Since the machine in question is my router, I've set it up for a little forgiveness (otherwise there'd be too many false positives) and only lock out after 2 bad attempts. This allows some false negatives to pass and locks out the people hammering your IP stack. Also, Advanced Stealth Scan mode immediately locks out anyone sending impossible packets. As for the spoof bounced attack being trackable, I would say that the best hope to track down the offender would be to have an IDS running on the middle-man, since that's the only host that gets to see the true identity of the attacker.

MULTIPLE CHOICE TEST QUESTION…
What is the expected response to a SYN-FIN impossible packet from an open port and a closed port?

A. RESET-ACK from an open port, RESET from a closed port.
A. SYN-ACK from an open port, RESET from a closed port.
B. FIN from an open port, RESET from a closed port.
C. SYN from an open port, SYN-ACK from a closed port.
Answer: B

DETECT NUMBER 2: SUBSEVEN TROJAN SCANS

Tcpdump of SubSeven Scans (format) (Source—HOME OFFICE):

November 16, 2000:

```
00:38:47.556229 24.13.4.120.4474 > mylinuxgw.homeoffice.net.27374: S 3612789506:3612789506(0)
win 16384 <mss 1460,nop,nop,sackOK> (DF)
00:38:50.370103 24.13.4.120.4474 > mylinuxgw.homeoffice.net.27374: S 3612789506:3612789506(0)
win 16384 <mss 1460,nop,nop,sackOK> (DF)
15:17:55.081378 24.5.88.72.2903 > mylinuxgw.homeoffice.net.27374: S 15644664:15644664(0) win
65535 <mss 1460,nop,nop,sackOK> (DF)
15:17:58.023728 24.5.88.72.2903 > mylinuxgw.homeoffice.net.27374: S 15644664:15644664(0) win
65535 <mss 1460,nop,nop,sackOK> (DF)
19:49:33.716113 24.64.139.94.3254 > mylinuxgw.homeoffice.net.27374: S 10172427:10172427(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
19:49:45.997480 24.64.139.94.3254 > mylinuxgw.homeoffice.net.27374: S 10172427:10172427(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
```

November 19, 2000:

```
05:19:54.640827 24.71.146.83.4629 > mylinuxgw.homeoffice.net.27374: S 29261549:29261549(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
14:37:26.606596 24.8.96.38.2690 > mylinuxgw.homeoffice.net.27374: S 28590945:28590945(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
```

```
14:37:29.548761 24.8.96.38.2690 > mylinuxgw.homeoffice.net.27374: S 28590945:28590945(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
17:03:36.262487 204.26.95.21.2904 > mylinuxgw.homeoffice.net.1243: S 3826312593:3826312593(0)
win 16384 <mss 1460,nop,nop,sackOK> (DF)
```

November 20, 2000:

```
03:59:06.159863 24.247.4.196.2653 > mylinuxgw.homeoffice.net.27374: S 912734795:912734795(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
04:47:50.847468 24.2.169.39.3901 > mylinuxgw.homeoffice.net.27374: S 38826643:38826643(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
11:44:14.138070 24.2.250.182.4236 > mylinuxgw.homeoffice.net.1243: S 89468090:89468090(0) win
8192 <mss 1460,nop,nop,sackOK>
11:44:17.104804 24.2.250.182.4236 > mylinuxgw.homeoffice.net.1243: S 89468090:89468090(0) win
8192 <mss 1460,nop,nop,sackOK>
11:44:23.123319 24.2.250.182.4236 > mylinuxgw.homeoffice.net.1243: S 89468090:89468090(0) win
8192 <mss 1460,nop,nop,sackOK>
11:44:35.164010 24.2.250.182.4236 > mylinuxgw.homeoffice.net.1243: S 89468090:89468090(0) win
8192 <mss 1460,nop,nop,sackOK>
19:56:53.173599 24.14.222.248.1059 > mylinuxgw.homeoffice.net.27374: S 32981117:32981117(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
20:02:13.437914 24.7.62.184.2062 > mylinuxgw.homeoffice.net.27374: S 5408811:5408811(0) win 8192
<mss 1460,nop,nop,sackOK> (DF)
```

Associated SYSLOG entries from IPChains (format):

```
Nov 16 00:38:47 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.13.4.120:4474
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=6050 F=0x4000 T=119
Nov 16 00:38:50 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.13.4.120:4474
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=6154 F=0x4000 T=119
Nov 16 15:17:55 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.5.88.72:2903
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=32339 F=0x4000 T=52
Nov 16 15:17:58 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.5.88.72:2903
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=3924 F=0x4000 T=52
Nov 16 19:49:33 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.64.139.94:3254
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=132 F=0x4000 T=117
Nov 16 19:49:45 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.64.139.94:3254
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=6021 F=0x4000 T=117

Nov 19 05:19:54 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.71.146.83:4629
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=53118 F=0x4000 T=112
Nov 19 14:37:26 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.8.96.38:2690
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=18558 F=0x4000 T=118
Nov 19 14:37:29 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.8.96.38:2690
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=64382 F=0x4000 T=118
Nov 19 17:03:36 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 204.26.95.21:2904
mylinuxgw.homeoffice.net:1243 L=48 S=0x00 I=43440 F=0x4000 T=112

Nov 20 03:59:06 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.247.4.196:2653
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=22703 F=0x4000 T=114
Nov 20 04:47:50 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.2.169.39:3901
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=29425 F=0x4000 T=120
Nov 20 11:44:14 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.2.250.182:4236
mylinuxgw.homeoffice.net:1243 L=48 S=0x00 I=59144 F=0x0000 T=23
Nov 20 11:44:17 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.2.250.182:4236
mylinuxgw.homeoffice.net:1243 L=48 S=0x00 I=6153 F=0x0000 T=23
Nov 20 11:44:23 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.2.250.182:4236
mylinuxgw.homeoffice.net:1243 L=48 S=0x00 I=28937 F=0x0000 T=23
Nov 20 11:44:35 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.2.250.182:4236
mylinuxgw.homeoffice.net:1243 L=48 S=0x00 I=60681 F=0x0000 T=23
Nov 20 19:56:53 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.14.222.248:1059
```

```
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=50275 F=0x4000 T=118
Nov 20 20:02:13 C405122-A kernel: Packet log: input DENY eth0 PROTO=6 24.7.62.184:2062
mylinuxgw.homeoffice.net:27374 L=48 S=0x00 I=19138 F=0x4000 T=120
```

Note that above my IP is bolded, and I've highlighted each distinct attacker in red. That's 11 different attacking hosts in a span of 5 days.

SPOOFED? …

Here we see many different hosts' addresses scanning my machine for known ports used by SubSeven. I would normally assume that someone doing reconnaissance like this would have a non-spoofed address somewhere in this mess of packets. However, as we saw in the first scan it is possible now to scan using spoofed addresses by bouncing the packets off of a middle-man, so to speak. For a detailed description on this technique, refer to the Attack Mechanism section in detect 1. Now our job is much more difficult in discerning whether a source is spoofed our not. It is entirely possible to apply the technique mentioned earlier to any tcp port, so tcp/1243 and tcp/27374 would be no exception. I was able to do some counter-intelligence on the machines that the source addresses above point to, however. The table below sums up my results:

Host	Response to Hping –S (or –SW to discern Windows vs. *NIX)
24.13.4.120	No Response
24.5.88.72	No Response
204.26.95.21	No Response
24.2.169.39	ICMP Host unreachable returned from 24.7.74.66
24.7.62.184	No Response
24.64.139.94	Alive but busy
24.2.250.182	QUIET—Windows (used hping –SW)
24.71.146.83	QUIET—Windows (used hping –SW)
24.8.96.38	QUIET—Windows (used hping –SW)
24.247.4.196	QUIET—*NIX (used hping –S)
24.14.222.248	QUIET—Windows (used hping –SW)

This table yields some interesting results. I can't conclude for certain, but if these machines traffic patterns were always like this (which is a bad assumption so I'm not going to jump to it) then we can see that the last 5 machines in this list would be potential middle-men for this type of spoof bounce attack (at least they would be at the late hour of night that I'm writing this). However, without some correlation, or this person trying to use me as their middle-man (and therefore I'd catch them on my IDS), it's VERY hard to say if the address is spoofed. I'm not sure it can be discerned without corollary evidence with these machines.

DESCRIPTION OF ATTACK…

In this case the attack was limited to searching for machines infected with the SubSeven Trojan. The goal here is to find any machines that are listening on tcp ports 1243 or 27374. It is only when the attacker gets a hit that the attack gets truly interesting. At that point the attacker would be able to do anything to my box that they wanted to do. To quote the Symantec website on its abilities (http://www.symantec.com/avcenter/venc/data/sub.seven.20.html), "Some of the capabilities include searching/retrieving/sending files, stealing passwords, changing the colors/resolution, playing sounds and changing the date/time". This pretty much enables the attacker to own you. Here's a network dump of SubSeven in action :

(note: This is a recreation of the attack on a protected network. This is not a trace collected "in the wild")

SUCCESSFUL NETWORK SCAN FOR PORT 27374 (done using Snort… format):

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
11/22-09:45:29.171480 sub7.attacker.net:2686 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:1270  DF
******S* Seq: 0x4632C39   Ack: 0x0   Win: 0x2000
TCP Options => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
11/22-09:45:29.171608 unsuspecting.victim.org:27374 -> sub7.attacker.net:2686
TCP TTL:128 TOS:0x0 ID:6656  DF
***A**S* Seq: 0x89E99   Ack: 0x4632C3A   Win: 0x2238
TCP Options => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
11/22-09:45:29.172988 sub7.attacker.net:2686 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:1526  DF
***A**** Seq: 0x4632C3A   Ack: 0x89E9A   Win: 0x2238
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
11/22-09:45:29.175053 unsuspecting.victim.org:27374 -> sub7.attacker.net:2686
TCP TTL:128 TOS:0x0 ID:6912  DF
***AP*** Seq: 0x89E9A   Ack: 0x4632C3A   Win: 0x2238
63 6F 6E 6E 65 63 74 65 64 2E 20 74 69 6D 65 2F  connected. time/
64 61 74 65 3A 20 30 38 3A 34 35 2E 33 33 20 2D  date: 08:45.33 -
20 4E 6F 76 65 6D 62 65 72 20 32 32 2C 20 32 30   November 22, 20
30 30 2C 20 57 65 64 6E 65 73 64 61 79 2C 20 76  00, Wednesday, v
65 72 73 69 6F 6E 3A 20 42 6F 4E 75 53 20 32 2E  ersion: BoNuS 2.
31                                               1
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
11/22-09:45:29.351582 sub7.attacker.net:2686 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:2550  DF
***A**** Seq: 0x4632C3A   Ack: 0x89EEB   Win: 0x21E7
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

Note in the section of Snort log, we see the attacker sending a SYN to port 27374 on the target box. Then, since SubSeven is actually installed on this machine, it replies with a SYN-ACK. The attacker's box replies with an ACK. The victim then replies with an ACK-PUSH to send the attacker the connection time and version of SubSeven server installed on the infected machine. The attacker's machine then replies to this ACK-PUSH with an ACK. I've cut out the FIN's at the end of the conversation above to keep things brief. Now let's look at the actual session replay:

(note: This is a recreation of the attack on a protected network. This is not a trace collected "in the wild")

SAMPLE SESSION:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:30.511204 sub7.attacker.net:2750 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:20214  DF
******S* Seq: 0x4633156   Ack: 0x0   Win: 0x2000
TCP Options => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:30.511314 unsuspecting.victim.org:27374 -> sub7.attacker.net:2750
TCP TTL:128 TOS:0x0 ID:7680  DF
***A**S* Seq: 0x8A3D5   Ack: 0x4633157   Win: 0x2238
TCP Options => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:30.511432 sub7.attacker.net:2750 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:20470  DF
***A**** Seq: 0x4633157   Ack: 0x8A3D6   Win: 0x2238
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:30.513245 unsuspecting.victim.org:27374 -> sub7.attacker.net:2750
TCP TTL:128 TOS:0x0 ID:7936  DF
***AP*** Seq: 0x8A3D6   Ack: 0x4633157   Win: 0x2238
63 6F 6E 6E 65 63 74 65 64 2E 20 74 69 6D 65 2F  connected. time/
64 61 74 65 3A 20 30 38 3A 34 35 2E 33 34 20 2D  date: 08:45.34 -
20 4E 6F 76 65 6D 62 65 72 20 32 32 2C 20 32 30   November 22, 20
30 30 2C 20 57 65 64 6E 65 73 64 61 79 2C 20 76  00, Wednesday, v
65 72 73 69 6F 6E 3A 20 42 6F 4E 75 53 20 32 2E  ersion: BoNuS 2.
31                                               1
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:30.653575 sub7.attacker.net:2750 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:20726  DF
***A**** Seq: 0x4633157   Ack: 0x8A427   Win: 0x21E7
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:45.408800 sub7.attacker.net:2750 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:29431  DF
***AP*** Seq: 0x4633157   Ack: 0x8A427   Win: 0x21E7
4D 57 3A 35 33 5A 4A 58 58 59 6F 75 20 6A 75 73  MW:53ZJXXYou jus
74 20 67 6F 74 20 6F 77 6E 65 64 2E 2E 2E        t got owned...
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:45.553037 unsuspecting.victim.org:27374 -> sub7.attacker.net:2750
TCP TTL:128 TOS:0x0 ID:8192  DF
***A**** Seq: 0x8A427   Ack: 0x4633175   Win: 0x221A
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 11/22-09:45:48.926343 unsuspecting.victim.org:27374 -> sub7.attacker.net:2750
TCP TTL:128 TOS:0x0 ID:8448  DF
***AP*** Seq: 0x8A427   Ack: 0x4633175   Win: 0x221A
75 73 65 72 20 63 6C 69 63 6B 65 64 20 3A 20 59  user clicked : Y
65 73 2E                                         es.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:49.081446 sub7.attacker.net:2750 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:30711  DF
***A**** Seq: 0x4633175   Ack: 0x8A43A   Win: 0x21D4
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/22-09:45:54.747756 sub7.attacker.net:2750 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:46327  DF
***A***F Seq: 0x4633175   Ack: 0x8A43A   Win: 0x21D4
```

```
 =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+:
 11/22-09:45:54.747870 unsuspecting.victim.org:27374 -> sub7.attacker.net:2750
TCP TTL:128 TOS:0x0 ID:8704  DF
***A**** Seq: 0x8A43A   Ack: 0x4633176   Win: 0x221A
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
11/22-09:45:54.749416 unsuspecting.victim.org:27374 -> sub7.attacker.net:2750
TCP TTL:128 TOS:0x0 ID:8960  DF
***A***F Seq: 0x8A43A   Ack: 0x4633176   Win: 0x221A
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
11/22-09:45:54.749588 sub7.attacker.net:2750 -> unsuspecting.victim.org:27374
TCP TTL:128 TOS:0x0 ID:46583  DF
***A**** Seq: 0x4633176   Ack: 0x8A43B   Win: 0x21D4
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

Here we see the first three packets representing the standard tcp 3-way handshake between an ephemeral port (2750) on sub7.attacker.net and the SubSeven port of 27374 on unsuspecting.victim.org. The fourth frame we see is the pushing of the connection time and server version from the victim to the attacker (as seen in the scan previously discussed), and the corresponding ACK from the attacker. The only action taken on the victim in this session was the sending of a popup message to the victim's display, stating "You just got owned…". We see this text in the 6th frame in the sample above. The 7th frame is the victim acknowledging the receipt of that message. The 8th frame actually sends back the user's response to the attacker. Note the ASCII "user clicked: Yes" in the 8th frame. The 9th frame is the attacker's box acknowledging receipt of the user's input, and the rest of the sample is tearing down the connection.

ATTACK MECHANISM…

The mechanism of attack in the live detects here is simple. The attacker is simply using a tool built into the SubSeven client to scan network address ranges for a given port. As we see in the detect, the server commonly runs on 1243 or 27374. However, this port number is configurable by the sender of the original Trojan. Therefore, if you've been specifically targeted by a potential hacker who sends this Trojan to you, that person can make the port be any number they want (assuming there's not something already bound to that port). Then, since they set the port number, they will know the port to look for in the future. The scan for the SubSeven server simply requires sending a tcp SYN packet to the port in question. If the attacker gets a SYN-ACK back, then they own you. If not, then there's no SubSeven server there.

CORRELATIONS…

Unfortunately, the SubSeven Trojan has become widespread enough that the two destination ports in my detect above have now become infamous. Plugging these numbers into the port search engine at http://www.snort.org/Database/portsearch.asp will verify this.

EVIDENCE OF ACTIVE TARGETING?…

The fact that all the live packets detected here were bound for the two commonly used ports of SubSeven would insinuate that this was NOT active targeting. It is clearly the work of script kiddies. However, if the traffic were detected going to a port other than these two ports, it would be a case of active targeting, since the attacker would know the "unusual" port settings used when setting up the Trojan before distribution. The same attacker would need to be the one (or someone the attacker told) who would be knocking on those strange ports. Things to set your IDS or stateful firewall to look for besides these two ports are discussed in the Defensive Recommendation section below.

SEVERITY…

Criticality: The target, once again, was my gateway box. Therefore, this attack could be classified as a ***5***.

Lethality: The attack was not lethal at all, since SubSeven is a Windows Trojan and my gateway box is Linux. Also, the packets were filtered before hitting my TCP stack anyway. Therefore these factors lead to a ***1***.

Host-based Countermeasures: The host based countermeasures in place (Portsentry by Psionic) didn't even get a chance to fire, but if the attacker did hit my box, since there's nothing running on that port, Portsentry would have locked them out after2 attempts (set to 2 to reduce false positives since there's internet traffic hitting this box). Therefore, I'd give this category a ***4***.

Network Countermeasures: The IPChains input ruleset stopped this packet before it even got to my tcp stack. Also, the output ruleset stopped any host or port unreachable messages, effectively silencing any response whatsoever to the would-be attacker. Therefore, I'd give this category a ***5***.

Severity = (Criticality + Lethality) – (Host-based Countermeasures + Network Countermeasures) = (5 + 1) – (4 + 5) = -3.

DEFENSIVE RECOMMENDATION…

The obvious defense against the script kiddies here is to make sure you've at least got a packet filtering firewall up that doesn't allow any traffic bound for destination port 27374 or 1243. Beyond that, it's clearly a good idea to no allow traffic from any random stranger on the internet into your network at all. However, since packet filtering firewalls are easily defeated, having an IDS (like Snort) set up to watch for these destination ports is also a good idea. Another thing for Snort (or a stateful firewall or application gateway firewall) to watch for is any noticeable strings in the application layer of these packets. One thing that stood out to me in the sample sessions and scans above is to prevent packets with the string "BoNuS" in the payload. This is clearly not a common (case sensitivity included) string to be showing up in your packets. Also, at the point when this packet shows up the 3-way handshake between the attacker and your victimized box has taken place already. However, it would be a good idea to watch for it with your IDS, since it would cover situations where the attacker changed the port. It is very possible that this string would not be in other versions of this Trojan, but every little bit helps when we're setting up our network defenses.

MULTIPLE CHOICE TEST QUESTION…

What are the 2 most commonly used ports for the Trojan SubSeven?

A. tcp/31337 and tcp/10067.
B. tcp/12361 and udp/12361
C. tcp/1243 and tcp/27374
D. udp/1243 and udp/12346

answer: c

DETECT NUMBER 3: FULL BLOWN PORT SCAN

Tcpdump of full port scan (Source—HOME OFFICE) (format):
Date: 11/18/2000

```
1.  04:39:24.713901 210.111.4.168.23 > mylinuxgw.homeoffice.net.23: . ack 2099606894 win 1028 (ttl 28,
id 39426)
2. 04:39:24.713991 mylinuxgw.homeoffice.net.23 > 210.111.4.168.23: R 2099606894:2099606894(0) win
0 (ttl 255, id 36400)
3. 04:39:24.827317 210.111.4.168.25 > mylinuxgw.homeoffice.net.25: . ack 2099606894 win 1028 (ttl 28,
id 39426)
4. 04:39:24.827358 mylinuxgw.homeoffice.net.25 > 210.111.4.168.25: R 2099606894:2099606894(0) win
0 (ttl 255, id 36401)
5. 04:39:24.941886 210.111.4.168.143 > mylinuxgw.homeoffice.net.143: . ack 2099606894 win 1028 (ttl
28, id 39426)
6. 04:39:24.941924 mylinuxgw.homeoffice.net.143 > 210.111.4.168.143: R 2099606894:2099606894(0)
win 0 (ttl 255, id 36402)
7. 04:39:25.055485 210.111.4.168.110 > mylinuxgw.homeoffice.net.110: . ack 2099606894 win 1028 (ttl
28, id 39426)
8. 04:39:25.055534 mylinuxgw.homeoffice.net.110 > 210.111.4.168.110: R 2099606894:2099606894(0)
win 0 (ttl 255, id 36403)
9. 04:39:25.169744 210.111.4.168.80 > mylinuxgw.homeoffice.net.80: . ack 2099606894 win 1028 (ttl 27,
id 39426)
10. 04:39:25.169787 mylinuxgw.homeoffice.net.80 > 210.111.4.168.80: R 2099606894:2099606894(0)
win 0 (ttl 255, id 36404)
11. 04:39:25.572222 210.111.4.168.1524 > mylinuxgw.homeoffice.net.23: S 772688882:772688882(0)
win 32120 <mss 1460,sackOK,timestamp 61625096[|tcp]> (DF) (ttl 50, id 32800)
12. 04:39:25.573003 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61625096[|tcp]>   (DF) (ttl 50, id 32801)
13. 04:39:28.564334 210.111.4.168.1524 > mylinuxgw.homeoffice.net.23: S 772688882:772688882(0)
win 32120 <mss 1460,sackOK,timestamp 61625396[|tcp]>   (DF) (ttl 50, id 32841)
14. 04:39:28.564414 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61625396[|tcp]> (DF) (ttl 50, id 32842)
15. 04:39:34.564106 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61625996[|tcp]>   (DF) (ttl 50, id 32920)
16. 04:39:46.564094 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61627196[|tcp]>   (DF) (ttl 50, id 33079)
17. 04:40:10.564610 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61629596[|tcp]> (DF) (ttl 50, id 33433)
18. 04:40:58.611725 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61634396[|tcp]> (DF) (ttl 50, id 33770)
19. 04:44:34.570971 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61655996[|tcp]> (DF) (ttl 50, id 35695)
20. 04:46:34.574315 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61667996[|tcp]> (DF) (ttl 50, id 36366)
21. 04:48:34.575551 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61679996[|tcp]> (DF) (ttl 50, id 36838)
22. 04:50:34.578619 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61691996[|tcp]> (DF) (ttl 50, id 37250)
23. 04:52:34.581400 210.111.4.168.1525 > mylinuxgw.homeoffice.net.111: S 775720009:775720009(0)
win 32120 <mss 1460,sackOK,timestamp 61703996[|tcp]> (DF) (ttl 50, id 38169)
24. 04:52:34.581785 210.111.4.168.4167 > mylinuxgw.homeoffice.net.23: S 1588284527:1588284527(0)
win 32120 <mss 1460,sackOK,timestamp 61703996[|tcp]> (DF) (ttl 50, id 38170)
25. 04:52:37.581242 210.111.4.168.4167 > mylinuxgw.homeoffice.net.23: S 1588284527:1588284527(0)
win 32120 <mss 1460,sackOK,timestamp 61704296[|tcp]> (DF) (ttl 50, id 38197)
26. 04:52:37.581474 210.111.4.168.1 > mylinuxgw.homeoffice.net.23: S 1945975380:1945975380(0) ack
294106719 win 1028 (ttl 28, id 39426)
27. 04:52:37.581569 mylinuxgw.homeoffice.net.23 > 210.111.4.168.1: R 294106719:294106719(0) win 0
(ttl 255, id 36497)
28. 04:52:37.600692 210.111.4.168.2 > mylinuxgw.homeoffice.net.23: F 1640519008:1640519008(0) win
1028 (ttl 28, id 39426)
```

```
29. 04:52:37.600758 mylinuxgw.homeoffice.net.23 > 210.111.4.168.2: R 0:0(0) ack 1640519008 win 0 (ttl
255, id 36498)
30. 04:52:37.621272 210.111.4.168.3 > mylinuxgw.homeoffice.net.23: F 1640519008:1640519008(0) ack
105223976 win 1028 (ttl 28, id 39426)
31. 04:52:37.621330 mylinuxgw.homeoffice.net.23 > 210.111.4.168.3: R 105223976:105223976(0) win 0
(ttl 255, id 36499)
32. 04:52:37.640791 210.111.4.168.4 > mylinuxgw.homeoffice.net.23: SF 1640519008:1640519008(0)
win 1028 (ttl 28, id 39426)
33. 04:52:37.660849 210.111.4.168.5 > mylinuxgw.homeoffice.net.23: P win 1028 (ttl 28, id 39426)
34. 04:52:37.660919 mylinuxgw.homeoffice.net.23 > 210.111.4.168.5: R 0:0(0) ack 1640519008 win 0 (ttl
255, id 36500)
35  04:52:43.582987 210.111.4.168.4167 > mylinuxgw.homeoffice.net.23: S 1588284527:1588284527(0)
win 32120 <mss 1460,sackOK,timestamp 61704896[|tcp]> (DF) (ttl 50, id 38261)
36. 04:52:43.782003 210.111.4.168.4206 > mylinuxgw.homeoffice.net.1: S 1597130314:1597130314(0)
win 32120 <mss 1460,sackOK,timestamp 61704916[|tcp]> (DF) (ttl 50, id 38262)
37. 04:52:46.781224 210.111.4.168.4206 > mylinuxgw.homeoffice.net.1: S 1597130314:1597130314(0)
win 32120 <mss 1460,sackOK,timestamp 61705216[|tcp]> (DF) (ttl 50, id 38277)
```

(note: I added the bolded numbers above to make the lines more easily referenced from the discussion below.)

Associated IPChains SYSLOG entries (format):

```
Nov 18 04:39:25 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1524 mylinuxgw.homeoffice.net:23 L=60 S=0x00 I=32800 F=0x4000 T=50
Nov 18 04:39:25 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=32801 F=0x4000 T=50
Nov 18 04:39:28 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
 210.111.4.168:1524 mylinuxgw.homeoffice.net:23 L=60 S=0x00 I=32841 F=0x4000 T=50
Nov 18 04:39:28 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=32842 F=0x4000 T=50
Nov 18 04:39:34 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=32920 F=0x4000 T=50
Nov 18 04:39:46 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=33079 F=0x4000 T=50
Nov 18 04:40:10 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=33433 F=0x4000 T=50
Nov 18 04:40:58 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=33770 F=0x4000 T=50
Nov 18 04:44:34 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=35695 F=0x4000 T=50
Nov 18 04:46:34 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=36366 F=0x4000 T=50
Nov 18 04:48:34 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=36838 F=0x4000 T=50
Nov 18 04:50:34 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=37250 F=0x4000 T=50
Nov 18 04:52:34 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:1525 mylinuxgw.homeoffice.net:111 L=60 S=0x00 I=38169 F=0x4000 T=50
Nov 18 04:52:34 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:4167 mylinuxgw.homeoffice.net:23 L=60 S=0x00 I=38170 F=0x4000 T=50
Nov 18 04:52:37 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:4167 mylinuxgw.homeoffice.net:23 L=60 S=0x00 I=38197 F=0x4000 T=50
Nov 18 04:52:37 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:4 mylinuxgw.homeoffice.net:23 L=40 S=0x00 I=39426 F=0x0000 T=28
Nov 18 04:52:43 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:4167 mylinuxgw.homeoffice.net:23 L=60 S=0x00 I=38261 F=0x4000 T=50
Nov 18 04:52:43 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
```

```
210.111.4.168:4206 mylinuxgw.homeoffice.net:1 L=60 S=0x00 I=38262 F=0x4000 T=50
Nov 18 04:52:46 C405122-A kernel: Packet log: input DENY eth0 PROTO=6
210.111.4.168:4206 mylinuxgw.homeoffice.net:1 L=60 S=0x00 I=38277 F=0x4000 T=50
```

Portsentry SYSLOG entries associated (format):

```
Nov 18 04:52:42 mylinuxgw portsentry[868]: attackalert: Unknown Type: Packet
Flags: SYN: 0 FIN: 0 ACK: 0 PSH: 1 URG: 0 RST: 0 from host:
210.111.4.168/210.111.4.168 to TCP port: 23
Nov 18 04:52:42 mylinuxgw portsentry[868]: attackalert: Host 210.111.4.168 has
been blocked via wrappers with string: "ALL: 210.111.4.168"
Nov 18 04:52:42 mylinuxgw portsentry[868]: attackalert: Host 210.111.4.168 has
been blocked via dropped route using command: "/sbin/ipchains -I input 1 -s
210.111.4.168 -j DENY -l"
```

SPOOFED? …

The traces listed above show some interesting things. First of all, the first 10 lines of the tcpdump trace show an interesting IP ID. See the 39426 there? Also note the fact that all of these first 10 lines show identical source & destination ports. However, the flags set on packets incoming from that attacker are clearly not SYN-FIN, as in the first detect I discussed and the correlation by Terry Bidwell, but instead are set to ACK. Does this mean that Terry's theories about script kiddies having a doctored version of Idlescan is correct? If so, then the address is spoofed. We'll get to that later. The entire trace shows as coming from a single IP address. Therefore it's definitely not the classic "flood the victim with spoofed IP's and hope they don't notice my real IP in there" mentality in the attacker's mind. Also, the attacker definitely fits the current trend of IP ID 39426 with src port = dst port used in the new "Spoof Bounced" scans. Note, though, that after these first 10 lines of tcpdump the attacker switches gears. Now he's trying to send SYN packets to my telnet port and my portmapper. The attack still shows up as that same IP address, though. Note now (tcpdump lines 11-25 above) that the attacker's IP ID's are no longer consistently 39426, nor are the source and destination ports identical. This suggests that there may be holes in Terry Bidwell's theory about script kiddies. It could be that this attacker was using the theorized Idlescan variant in the first 10 lines, and then switched to something like hping2 for the next 15 lines. However, from lines 26 to the end the attacker switches back to using IP ID 39426. But wait! (S)He is not using matching ports this time!

Another interesting thing to note is the TTL associated with each entry. In the first section, where we have IP ID on incoming packets equal to 39426 and source port = destination port, the TTL is always 28. However, in the middle section, when the attacker is sending SYN only packets, the TTL's change to 50. Then, when the attacker starts sending impossible packets near the end (FIN only, PUSH only, SYN-FIN) the TTL changes back to 28. This would imply the possibility that the first section (lines 1-10) are spoof bounced, then 11-25 are not spoofed, then lines 26-37 are spoofed again.

DESCRIPTION OF ATTACK…

In lines 1-10 of the tcpdump above the attacker attempts to do something to my box that begins with sending me an ACK. By the rest of the packet signature (IP ID=39426, source port = destination port), it looks like an attempt at a "Spoof Bounced" scan. However, the ACK flag alone will not work with this technique, as will be discussed in the Attack Mechanism section. My box simply sends RESETs to all 5 of his replies.

Then in lines 11-25 of the attempted attack, the attacker switches gears. (S)He is now sending SYN packets to my telnet port (tcp/23) and my portmapper (tcp/111). Note that the attacker has forgone the IP ID 39426 in this section, and the source and destination ports no longer match. These generate no responses at all from my box, since my firewall has blocked the packets. Note the IPChains SYSLOG entries in the excerpt directly after the tcpdump above. The first entry here is when the attacker has switched gears and gone for the SYNs. IPChains let in the ACK packets (due to the fact that it's not stateful and is set up to allow established traffic), so it doesn't start complaining until line 11 of the tcpdump traffic. At this point, IPChains stops each of these packets from reaching my box (and I have telnet and portmapper disabled, anyway). However, no outbound traffic is seen either because the output chain of the IPChains configuration blocks ICMP unreachables. After these failed attempts, the attacker (on line 26) tries to send me a SYN-ACK from source port 1 to destination port 23, which the firewall lets through because it's not a SYN only packet, and my box replies with a RESET because I didn't send him a SYN. Now, frustrated, the attacker attempts a FIN from source port 2 to source port 23 (line 28 above). My box replies with a RESET-ACK(line 29). In line 30, the attacker is seen sending me a FIN-ACK. I respond with a RESET in line 31. In line 32, the attacker sends a SYN-FIN to me, which is blocked by the firewall. Then in line 33 the attacker sends a PUSH to port 23, which is allowed through the firewall and generates a RESET-ACK in response. In lines 35-37, the attacker sends 3 SYN only packets, all of which are blocked by the firewall. With all of these strange packets attempted in the last section (FIN only, PUSH only, SYN-FIN), I suspect attempted OS fingerprinting. We'll see in the Attack Mechanism section next.

ATTACK MECHANISM…

It was clear by this trace that the hacker in question really wanted to exploit my machine. It appeared as though the attacker was attempting to Spoof Bounce a port scan to me using the technique described in Detect 1. However, by using an ACK only packet, this will not work. First, look at the trace of a successful Spoof Bounce mentioned earlier. Note that the packet sent to the target, with the spoofed IP address of the middle-man as the source, causes a response packet to be sent back to the machine in the source address (the middle-man). If the Spoof Bouncing technique is to be successful, then this packet sent to the middle-man must, in turn, cause the middle man to send a packet back to the target (thereby causing the middle-man to expend 1 IP ID number). If the packet that the target sends to the middle-man doesn't cause a response from the middle-man back to the target, then no IP ID's will be spent and the result is similar to the way a closed port looks in the Spoof Bounce. Since an ACK causes my box to send a RESET back to the middle-man, and there is no response needed to a RESET, then the middle-man doesn't need to reply to anything and no IP ID's are spent. As for the possibility for OS fingerprinting that I mentioned in the Description of Attack section, it looks like it could be an unsuccessful attempt at OS fingerprinting. I say unsuccessful mainly for one reason: the responses my Linux box sent him, in this specific instance of attack, were no different than a Windows box would have produced (regardless of whether the Windows box's ports were open or closed). Note the following table:

Flags	Windows 98		Linux		IRIX		
	Open Port	Closed Port	Open Port	Closed Port	Open Port	Closed Port	Oper Port

A	R	R	R	**R**	R	R	R
F	RA	RA	No Resp.	**RA**	RA	RA	No Resp
FA	R	R	R	**R**	R	R	R
SF	SA	RA	SA	RA	A	RA	SA
P	RA	RA	No Resp.	**RA**	RA	RA	No Resp

The responses he got from my box are marked in red. To begin with, I sent him a RESET in response to an ACK. Looking at the table above, that does him no good. Next, I sent him a RESET-ACK in response to a FIN. That does him a little good. However, remember that he sent this packet to port 23. He probably did this assuming telnet would be open. If he ran on the assumption that it was an open port, then that leads him to believe that I've got either a Windows box or an IRIX box. However, since he never really figured out if port 23 was up, he can't be sure that is a good assumption. Therefore, the FIN I responded to did him no good. Next came the FIN-ACK. That generated a RESET in response, which, according to the table above, does him no good. The SYN-FIN I never responded to, so that does nothing for the attacker either. Finally, I responded with a RESET-ACK to a PUSH. This is the same as the open AND closed port responses for Windows 98, and IRIX, as well as the closed port response for Linux and Solaris. Once again, if the attacker mistakenly assumes the telnet daemon is running, then he'll be misled. If he correctly assumes it's closed, then he doesn't know what to think because they all send RESET-ACKs from closed ports in response to a PUSH.

CORRELATIONS…

Richard Bejtlich posted the following trace at
http://www.ussrback.com/docs/papers/IDS/intv2.txt It is listed in tcpdump format.

```
22:08:33.913622 cable.modem.net.23 > dns.one.org.23: . ack 499410217 win 1028 (ttl 30, id
22:08:33.915481 dns.one.org.23 > cable.modem.net.23: R 499410217:499410217(0) win 0 (tt
22:08:33.954076 cable.modem.net.23 > dns.two.org.23: . ack 499410217 win 1028 (ttl 0, id 3
22:08:33.955461 dns.two.org.23 > cable.modem.net.23: R 499410217:499410217(0) win 0 (tt
22:08:33.982753 cable.modem.net.143 > dns.one.org.143: . ack 499410217 win 1028 (ttl 30, i
22:08:33.983904 dns.one.org.143 > cable.modem.net.143: R 499410217:499410217(0) win 0
22:08:34.061121 cable.modem.net.143 > dns.two.org.143: . ack 499410217 win 1028 (ttl 30, i
22:08:34.064069 dns.two.org.143 > cable.modem.net.143: R 499410217:499410217(0) win 0
22:08:39.161874 cable.modem.net.1146 > dns.two.org.23: S 2585837477:2585837477(0) win
22:08:39.170887 cable.modem.net.1147 > dns.two.org.143: S 2585589580:2585589580(0) wi
22:08:39.182221 cable.modem.net.1149 > dns.two.org.111: S 2583756762:2583756762(0) wi
22:08:39.183010 cable.modem.net.1151 > dns.two.org.79: S 2588862233:2588862233(0) win
22:08:39.190551 cable.modem.net.1152 > dns.two.org.53: S 2590571910:2590571910(0) win
22:08:39.212060 cable.modem.net.1153 > dns.two.org.31337: S 2585840391:2585840391(0)
22:08:39.224956 cable.modem.net.1157 > dns.two.org.21: S 2585906418:2585906418(0) win
22:08:39.261488 cable.modem.net.1162 > dns.one.org.23: S 2589761527:2589761527(0) win
22:08:39.264445 cable.modem.net.1163 > dns.one.org.143: S 2588328359:2588328359(0) wi
22:08:39.292663 cable.modem.net.1165 > dns.one.org.111: S 2590160130:2590160130(0) wi
22:08:39.305784 cable.modem.net.1167 > dns.one.org.79: S 2594918730:2594918730(0) win
22:08:39.307131 cable.modem.net.1168 > dns.one.org.53: S 2582494230:2582494230(0) win
22:08:39.307209 cable.modem.net.1169 > dns.one.org.31337: S 2590958670:2590958670(0)
22:08:39.344336 cable.modem.net.1173 > dns.one.org.21: S 2593455289:2593455289(0) win
```

Above we see behavior extremely similar to the first 2 sections of the detect I collected. The first section shows the use of the notorious IP ID 39426, with source port = destination port. Then the attacker switches gears and tries SYN connections to various ports with well known vulnerabilities. However, in my detect the attacker switched gears again and shot impossible packets at me. Note the changing of the TTL when the attacker switches to SYN connections only. The difference in TTL is very similar to my detect above. In fact, this detect and the analysis given by Richard are what made me think to check my TTLs. Here's Richard's analysis of the TTL issue:

> "We see a wide variance between the TTL 30 of stage one and TTL 52 of stage two. As these packets presumably come from the same host, we assume the tool generate the packets sets initially TTLs differently for each technique. Stage one shows IP id values each forged to be 39426. This may provide a signature clue for future encounters with this tool." --Richard Bejtlich, http://www.ussrback.com/docs/papers/IDS/intv2.txt

Here, Richard suggests that the TTL difference is due to different initial TTL values produced by different tools used in the attack. This is certainly possible. However, nowhere in this article does he mention the possibility of this being a Spoof Bounced attack. I will concede that the possibility that Richard suggests is just as likely as the first half of the traffic he noted being Spoof Bounced off of a middle-man, and then the 2nd half coming directly from that (compromised) middle-man.

EVIDENCE OF ACTIVE TARGETING?…

This is a VERY clear case of active targeting. The traffic cited in my detect (and the correlation above) is directed at gaining enough reconnaissance information to crack the box in question. Trying to connect many times to my telnetd and portmapper are dead giveaways of someone who wants into my box specifically. Also, the fact that there were impossible flag combinations involved proves malicious intent.

SEVERITY…

Criticality of Target: As discussed earlier, this box (mylinuxgw.homeoffice.net) is critical…***5***.

Lethality of Attack: The attacker was able to generate some responses to probing my box. However, none of the responses handed out seemed to be very helpful in the reconnaissance. Overall I'd rate this variable a ***2***.

Host-Based Countermeasures: This attacker was locked out by Portsentry, as seen in the beginning of this detect. However, this lockout did not come until the attacker sent the PUSH packet late in the attack. It would have fired in the first section of the attack, but the attacker only sent one packet to telnet, one to smtp, one to imap, one to pop3, and one to httpd. Since I have the settings on Portsentry set to 2 attempts, the attacker was able to sneak in unnoticed until sending the impossible packets. I would therefore have to rate this variable a ***3***.

Network-Based Countermeasures: These worked moderately well here. The IPChains did its job in stopping all the SYN packets sent in the second stage (tcpdump lines 11-25), but let through the ACK packets in the first stage and all but the SYN-FIN in the last stage. This is because the rule available in IPChains only specifies SYN only (or at least any SYN without an ACK). Therefore, it allows PUSH, FIN, FIN-ACK, and ACK through in this detect. I would therefore consider this variable a ***3***.

Adding up the totals in this section results in a severity of ***1***.

DEFENSIVE RECOMMENDATION…

It is clear that a better firewall would have been key in stopping this attack entirely. This hacker was able to do quite a bit right through the firewall that IPChains was responsible for. A stateful inspection firewall or application proxy gateway would not have let any of these packets through at all. Also, if the machine were not directly connected to the internet, tuning Portsentry not to be so forgiving about how many connections it takes to fire off IPChains blocking and tcpwrappers would be a good idea. However, if it is tuned down too much it could lead to denials of service.

MULTIPLE CHOICE TEST QUESTION…

What are the responses generated by open ports on a Linux box for the following tcp flag combinations (respectively)?

SYN, SYN-FIN, PUSH

A. SYN-ACK, RESET, RESET.
B. ACK, SYN-ACK, RESET-ACK.
C. SYN-ACK, SYN-ACK, RESET.
D. SYN-ACK, SYN-ACK, RESET-ACK.

Answer: D

DETECT NUMBER 4: ANOMALOUS WAN TRAFFIC

Tcpdump output (Source—HOME OFFICE) (format):

```
02:26:51.544676 wkstn1.officea.net > nonexist.officeb.net: icmp: echo request (ttl 31, id 23553)
02:27:50.129795 wkstn1.officea.net.1036 > private1.officeb.net.53:  1+ (41) (ttl 127, id 41730)
02:28:08.987243 wkstn1.officea.net.1045 > private1.officeb.net.53:  1+ (41) (ttl 127, id 59650)
02:28:09.485922 wkstn1.officea.net.1045 > nonexist.officeb.net.53:  1+ (41) (ttl 127, id 59906)
```

Cisco SYSLOG output (Source—COMPANY FOREIGN OFFICE)(format):

```
Nov 19 21:07:49 6X:router.officea.net 5372: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn2.officea.net(137) -> 190.10.92.2(137)
Nov 19 21:07:49 6X:router.officea.net 5373: 14w5d: %SEC-6-IPACCESSLOGDP: list 104 denied icmp
130.244.198.194 -> wkstn2.officea.net (3/1)
Nov 19 23:07:50 6X:router.officea.net 5423: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn2.officea.net(137) -> 190.10.92.2(137)
Nov 19 23:07:50 6X:router.officea.net 5424: 14w5d: %SEC-6-IPACCESSLOGDP: list 104 denied icmp
130.244.198.194 -> wkstn2.officea.net (3/1)
Nov 19 23:45:51 6X:router.officea.net 5443: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn2.officea.net(137) -> 190.10.92.2(137)
Nov 19 23:45:51 6X:router.officea.net 5444: 14w5d: %SEC-6-IPACCESSLOGDP: list 104 denied icmp
130.244.198.194 -> wkstn2.officea.net (3/1)
Nov 20 02:07:52 6X:router.officea.net 5510: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn2.officea.net(137) -> 190.10.92.2(137)
Nov 20 02:07:52 6X:router.officea.net 5511: 14w5d: %SEC-6-IPACCESSLOGDP: list 104 denied icmp
130.244.198.194 -> wkstn2.officea.net (3/1)
Nov 20 02:28:25 6X:router.officea.net 5529: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn1.officea.net(137) -> 190.100.92.2(137)
Nov 20 02:28:58 6X:router.officea.net 5532: 14w5d: %SEC-6-IPACCESSLOGDP: list 106 permitted icmp
```

```
wkstn1.officea.net -> nonexist.officeb.net (8/0)
Nov 20 02:29:57 6X:router.officea.net 5535: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn1.officea.net(1036) -> private1.officeb.net(53)
Nov 20 02:30:15 6X:router.officea.net 5536: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn1.officea.net(1045) -> private1.officeb.net(53)
Nov 20 02:30:16 6X:router.officea.net 5537: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn1.officea.net(1045) -> nonexist.officeb.net(53)
Nov 20 02:30:23 6X:router.officea.net 5538: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn1.officea.net(137) -> 190.100.92.2(137)
Nov 20 02:31:52 6X:router.officea.net 5544: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn2.officea.net(137) -> 190.10.92.2(137)
Nov 20 02:31:52 6X:router.officea.net 5545: 14w5d: %SEC-6-IPACCESSLOGDP: list 104 denied icmp
130.244.198.194 -> wkstn2.officea.net (3/1)
Nov 20 02:47:52 6X:router.officea.net 5556: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn2.officea.net(137) -> 190.10.92.2(137)
Nov 20 02:47:52 6X:router.officea.net 5557: 14w5d: %SEC-6-IPACCESSLOGDP: list 104 denied icmp
130.244.198.194 -> wkstn2.officea.net (3/1)
Nov 20 02:48:23 6X:router.officea.net 5559: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn1.officea.net(137) -> 190.100.92.2(137)
```

SPOOFED? ….

This traffic came in over the VPN from Windows NT Workstations in one office of my company to the protected internal addresses of my home office. One of the internal addresses that they hit was not even an existing host. The only box that knows anything about my internal addresses in my home office is the router. It is therefore strange that the workstations in the other office would be querying my private LAN addresses for DNS services at all. Also, they all use DHCP and I have verified that their DHCP assigned DNS servers are still correct. Note as well that they're not just querying DNS services. One of the workstations actually tried pinging a non-existent address on my private LAN here in my home office. The tcpdump trace above shows the traffic that my home office sniffer captured. That got me interested in what was going on over there in the other office. We don't have any sniffers set up there, yet, so the best I could do was to turn on all the logging I could on their router. The second list above is the result of that trace. The traffic highlighted in red is the traffic that corresponds to the tcpdump I have listed here. The time signatures are off by about 1 minute between the box that logged this Cisco traffic and the box that logged the tcpdump. I thought it best to leave it as captured (instead of editing the time signatures…=). I don't believe this traffic is spoofed, unless it's being spoofed from another machine on their LAN. I can safely make this assumption based on two reasons: (1) The router in the other office rejects incoming traffic with a source address of that LAN (spoof protection), and (2) all traffic coming from that LAN comes into my LAN via VPN. Any traffic coming from that LAN that is not over the VPN is dropped. Therefore I find it very unlikely that the traffic is spoofed.

DESCRIPTION OF ATTACK…

The traffic above fits two categories. The first category would include strange traffic coming from the 2 workstations on the other LAN to my LAN, and the second would be in pairs of packets. The pairs in the second category fit a strange pattern. First, the workstations in question on the other LAN send a udp/137 packet to machines outside the LAN. It's easy to overlook, but there are actually 2 machines they're sending to: 190.10.92.2 and 190.100.92.2. The second octet is different. Then an ICMP packet shows up and is blocked by the router and logged, coming from 130.244.198.194, destined for whatever box just sent a NetBIOS packet. This behavior had me stumped until I recently figured it out. At first it almost looked like someone trying unsuccessfully to learn the Spoof Bounce (recent Spoof Bounce on the brain…=). Then I realized by doing a tracert from an NT Server on that LAN that both of the 190. addresses were unreachable, and the last router to touch those packets was 130.244.198.194. That put my mind at ease. However, I'm still perplexed as to the reason these machines are trying to find DNS servers on my home office LAN. As far as I could find, there is no strange behavior programmed by Micro$oft to cause this to happen. I even found an article on Microsoft's website discussing the Microsoft DNS API. However, at one point, right after sending a NetBIOS broadcast to the unreachable address, wkstn1.officea.net tries to ping a non-existent host in my home office private address space. That is followed up by DNS queries to an existing and the same non-existent host in my home office private address space. Another perplexing issue is why on earth they are sending NetBIOS queries to those addresses to begin with. I have not seen any traffic coming in from those 190. addresses at all.

ATTACK MECHANISM…

It appears as though the Cisco log entries above suggest that my protected LAN is being scanned for DNS servers by workstations in the other office. My machine has not responded in the examples above because this has been happening for a couple of weeks now and Portsentry has automatically locked out the offending workstations. I'm leaving them locked out until I figure out what's going on here. It is REALLY difficult to tell what's going on here with only those Cisco log records, but that's the best I will do until my company sends me to set up a sniffer there. If it were really an attack, I would suspect that I would see attempts being made at other ports on my box, or other machines. However, this is not the case. I'm only seeing a 2-3 workstations in the other office sending what appear to be DNS requests to one of my hosts about 90% of the time, and the non-existent host about 10% of the time. The sample above is not necessarily representative of the frequency with which the addresses on my LAN are being hit. I chose this sample simply because there were multiple workstations in the other LAN and multiple addresses in my LAN involved in a short period of time, and they happened right in the middle of the mysterious udp packets to 190. addresses. Here's the tracert from the NT Server in the other office:

```
C:\>tracert 190.10.92.2
Tracing route to 190.10.92.2 over a maximum of 30 hops
  1   <10 ms   <10 ms   <10 ms  router.officea.net
  2    10 ms   <10 ms    10 ms  arouter.myisp.net
  3    10 ms    10 ms    10 ms  kst25.serial5-0c1.swip.net [130.244.5.125]
  4     *        *        *     Request timed out.
  5     *        *     ^C
C:\>
```

The same output results from a tracert to 190.100.92.2. Therefore, I think that there's a bug somewhere in the workstations involved. Another thing that strikes me as strange, though, is that there is another office in my company that has about 70 NT Workstations, and I'm not getting ANY traffic like this from the other office. I've turned on all the logging possible on the router, and I'm not seeing any other anomalous traffic of this sort (and there are only 6 workstations and 1 server in that office, so there's not that much traffic to read). I would suspect that someone has a backdoor, but as I stated, I'm not seeing that sort of traffic from the logs. Also, I strongly doubt that these users are saavy enough to start scanning my private LAN addresses for DNS servers. Therefore, I think this anomalous traffic is entirely made up of Microsoft bugs.

CORRELATIONS…
The only corollary evidence I can find regarding this behavior is a lack thereof. I have even found an article on Microsoft's developer site regarding their implementation of DNS, and there's nothing there either. Here's the URL I read:
http://www.microsoft.com/technet/winnt/reskit/sur_tcp2.asp
This tells me nothing, except that (in the words that made Microsoft infamous) it's probably not "by design"…=).

EVIDENCE OF ACTIVE TARGETING?…
If I thought that it looked malicious then I would strongly suspect active targeting. I say that it would be active targeting because there are two hosts being scanned for DNS servers on my network, and one of them doesn't even exist. There are no DNS servers in my private LAN. There are no packets coming from the third office I mentioned, which has 10 times the number of NT Workstations set up, and they are also using DHCP assigned DNS servers. However, since it looks like bugs to me, I'd have to theorize that it's probably not active targeting. Also, even if it were, the offending workstations are now locked out of my private LAN anyway.

SEVERITY…

Criticality of Target: The targets are both critical (my gateway box) and non-critical (non-existent). Therefore, to split the difference, we'll say ***3***.
Lethality of Attack: The attack (if it could be considered one) was ineffective at best, so I'd give this one a ***2***. (that high only because this one still makes me a little nervous).
Host-Based Countermeasures: The host that was an existing address has Portsentry installed and locked out the offending machines…***5***.
Network-Based Countermeasures: Portsentry has set up IPChains to keep these machines locked out. This occurred after the first 2 attempts, so the network based countermeasures seem to be working effectively. ***4***.
(4 + 2) – (5 + 4) = ***-3***

DEFENSIVE RECOMMENDATION…
It's hard to recommend defensive action against what looks to be a Microsoft bug (resistance is futile…). However, in case this is actually active targeting and a real attack, it is a good idea to have at *least* a packet filtering firewall, tcpwrappers, and Portsentry. I've found that these 3 items in combination really take out 90% of your network attackers.

MULTIPLE CHOICE TEST QUESTION…
Which of the following statements about the tcpdump trace and tracert below are NOT TRUE?
Cisco router logs, list 106 is the outbound ACL, list 104 is the inbound ACL.

```
Nov 19 21:07:49 6X:router.officea.net 5372: 14w5d: %SEC-6-IPACCESSLOGP: list 106 permitted udp
wkstn2.officea.net(137) -> 190.10.92.2(137)
Nov 19 21:07:49 6X:router.officea.net 5373: 14w5d: %SEC-6-IPACCESSLOGDP: list 104 denied icmp
```

130.244.198.194 -> wkstn2.officea.net (3/1)

```
C:\>tracert 190.10.92.2
Tracing route to 190.10.92.2 over a maximum of 30 hops
 1   <10 ms   <10 ms   <10 ms  router.officea.net
 2    10 ms   <10 ms    10 ms  arouter.myisp.net
 3    10 ms    10 ms    10 ms  kst25.serial5-0c1.swip.net [130.244.5.125]
 4     *        *        *     Request timed out.
 5     *        *     ^C
C:\>
```

A. ICMP TTL's are blocked inbound at arouter.myisp.net.
B. ICMP Host-Unreachables are blocked inbound at arouter.myisp.net.
C. UDP port 137 is allowed outbound at arouter.myisp.net
D. Traffic is dropping after the router 130.244.5.125.

Answer: A.

EVALUATE AN ATTACK

In this section I will evaluate the Spoof Bounced port scan. This scan is described in the SPOOFED_SCAN.txt file that comes with hping2-beta54. Examples of this technique in action are given in Detect 1, above. Since the basic workings are explained there I will not repeat them here. However, here I will evaluate which packet combinations will work with this attack. As we saw in Detect 1, the attack works with a SYN-FIN sent to the target host, with the middle-man's IP address as the spoofed source. As described in the hping HOWTO, a SYN will also work. I will give examples of the following flag combinations sent to open ports on a Windows 98 box: SYN, ACK, FIN, PUSH, URG, SYN-ACK, SYN-FIN, SYN-PUSH, SYN-URG, SYN-ACK-FIN, SYN-ACK-PUSH, SYN-ACK-URG, SYN-FIN-URG, SYN-FIN-PUSH, SYN-RESET, and SYN-FIN-RESET-PUSH-URG. I will not illustrate the effects on a closed port, because even when it works, it's not supposed to generate an affect on a closed port (that's how you tell it's closed…=).

SYN on an open port:

```
22:55:31.163118 target.box.net.2794 > middle.man.net.135: S 900756329:900756329(0) win 512
22:55:31.163525 middle.man.net.135 > target.box.net.2794: S 1489803:1489803(0) ack
900756330 win 8576 <mss 1460> (DF)
22:55:31.163583 target.box.net.2794 > middle.man.net.135: R 900756330:900756330(0) win 0
```

ACK on an open port:

```
22:55:41.101971 target.box.net.1588 > middle.man.net.135: . ack 1035253368 win 512
22:55:41.102169 middle.man.net.135 > target.box.net.1588: R 1035253368:1035253368(0) win 0
```

FIN on an open port:

```
22:55:50.975124 target.box.net.2743 > middle.man.net.135: F 559062598:559062598(0) win 512
22:55:50.975299 middle.man.net.135 > target.box.net.2743: R 0:0(0) ack 559062599 win 0
```

PUSH on an open port:

```
22:55:58.746118 target.box.net.2101 > middle.man.net.135: P win 512
22:55:58.746316 middle.man.net.135 > target.box.net.2101: R 0:0(0) ack 1227222676 win 0
```

URG on an open port:

```
22:56:05.905615 target.box.net.1859 > middle.man.net.135: . win 512 urg 0
22:56:05.905816 middle.man.net.135 > target.box.net.1859: R 0:0(0) ack 987405842 win 0
```

SYN-ACK on an open port:

```
22:56:12.373402 target.box.net.2990 > middle.man.net.135: S 1830994150:1830994150(0) ack
1853390452 win 512
22:56:12.373576 middle.man.net.135 > target.box.net.2990: R 1853390452:1853390452(0) win 0
```

SYN-FIN on an open port:

```
22:56:21.814945 target.box.net.2701 > middle.man.net.135: SF 142161700:142161700(0) win
512
22:56:21.815153 middle.man.net.135 > target.box.net.2701: S 1489819:1489819(0) ack
142161701 win 8576 <mss 1460> (DF)
22:56:21.815202 target.box.net.2701 > middle.man.net.135: R 142161701:142161701(0) win 0
```

SYN-PUSH on an open port:

```
22:56:30.945182 target.box.net.1456 > middle.man.net.135: SP 1692288021:1692288021(0) win
512
22:56:30.945396 middle.man.net.135 > target.box.net.1456: S 1489835:1489835(0) ack
1692288022 win 8576 <mss 1460> (DF)
22:56:30.945467 target.box.net.1456 > middle.man.net.135: R 1692288022:1692288022(0) win 0
```

SYN-URG on an open port:

```
22:56:41.510503 target.box.net.2624 > middle.man.net.135: S 1791934852:1791934852(0) win
512 urg 0
22:56:41.510721 middle.man.net.135 > target.box.net.2624: S 1489851:1489851(0) ack
1791934853 win 8576 <mss 1460> (DF)
22:56:41.510810 target.box.net.2624 > middle.man.net.135: R 1791934853:1791934853(0) win 0
```

SYN-ACK-FIN on an open port:

```
22:56:53.290074 target.box.net.3001 > middle.man.net.135: SF 1691449216:1691449216(0) ack
582718574 win 512
22:56:53.290271 middle.man.net.135 > target.box.net.3001: R 582718574:582718574(0) win 0
```

SYN-ACK-PUSH on an open port:

```
22:57:02.230021 target.box.net.2119 > middle.man.net.135: SP 1087061640:1087061640(0) ack
1063005720 win 512
22:57:02.230199 middle.man.net.135 > target.box.net.2119: R 1063005720:1063005720(0) win 0
```

SYN-ACK-URG on an open port:

```
22:57:13.470119 target.box.net.1902 > middle.man.net.135: S 108776347:108776347(0) ack
1775432598 win 512 urg 0
22:57:13.470315 middle.man.net.135 > target.box.net.1902: R 1775432598:1775432598(0) win 0
```

SYN-FIN-URG on an open port:

```
22:57:23.444747 target.box.net.2272 > middle.man.net.135: SF 386038223:386038223(0) win
512 urg 0
22:57:23.444956 middle.man.net.135 > target.box.net.2272: S 1489865:1489865(0) ack
386038224 win 8576 <mss 1460> (DF)
22:57:23.445059 target.box.net.2272 > middle.man.net.135: R 386038224:386038224(0) win 0
```

SYN-FIN-PUSH on an open port:

```
22:57:46.444357 target.box.net.1863 > middle.man.net.135: SFP 387648107:387648107(0) win
512
22:57:46.444571 middle.man.net.135 > target.box.net.1863: S 1489870:1489870(0) ack
387648108 win 8576 <mss 1460> (DF)
22:57:46.444668 target.box.net.1863 > middle.man.net.135: R 387648108:387648108(0) win 0
```

SYN-RESET on an open port:

```
        23:23:33.854757 target.box.net.2547 > middle.man.net.135: SR 1532982148:1532982148(0) win
512
```

SYN-FIN-RESET-PUSH-URG on an open port:

```
    23:23:41.163349 target.box.net.1135 > middle.man.net.135: SFRP 31074363:31074363(0) win
512 urg 0
```

I have highlighted the hits in red above. As we saw in the Attack Mechanism section in Detect 1, this technique begins with a quiet host on the internet to act as a middle-man. The quiet host, in response to tcp pings, will generate IP ID's that consistently increment by one. The attacker then sends a packet (with flags combinations in red above) to the target box, with this middle-man's IP address as the spoofed source address. As we see above, different flag combinations produce different responses from the target. All of these responses will be sent to the middle-man, since we spoofed his address. Some of these responses from the victim to the middle-man will generate responses by the middle-man. For instance, we all know that a 3-way handshake is initiated by a SYN from the client to the server, followed by a SYN-ACK response from the server to the client, followed by an ACK from the client to the server. In the case of this attack, the middle-man's IP is spoofed in the SYN. The server (the target in this case) sends a SYN-ACK. Now, since the client (middle-man) didn't send a SYN, then he has no idea where this SYN-ACK came from. Therefore the client (middle-man) sends a RESET to the server (target). BINGO! We just made the middle-man expend one of his IP ID's on the target machine, so the IP ID's we're still getting with our steady tcp ping will suddenly skip a number…gotcha! If the port is closed, then all packets sent to the target originally will generate either a RESET or a RESET-ACK. These two flag combinations will never cause the middle-man to respond in kind, so the closed ports will never make the middle-man waste an IP ID on the target.

As we see in the pattern above, the flag combinations that work are clear. Any packet that has a SYN and does not have an ACK or a RESET will work. We see that the RESET flag automatically overrides all other flags and causes no response at all. Also, judging by the behavior of the ACK flag, that appears to take second priority, since any combination of flags with the ACK bit set causes the target to respond with a RESET. Anything without a SYN set gets either a RESET or a RESET-ACK. Therefore, be creative with the flags in this attack! Just make sure that you've got a SYN and no ACK or RESET.

ANALYZE THIS!

SECURITY AUDIT OF GIAC ENTERPRISES

Performed by David Thibault, 11/22/2000.

This security audit was requested by GIAC Enterprises for the purpose of reviewing their security data set. The set is entirely made up of warnings generated by the Snort IDS. Due to difficulties arising from disk shortages, power outages, and the like, GIAC Enterprises (hereafter refered to as "the customer") has provided David Thibault (hereafter referred to as "the consultant") with a somewhat incomplete set of security logs. These logs are missing days of data, and are therefore not expected to give a complete picture of the customer's current security situation. Therefore the consultant's analysis should be considered a summary at best, not an all-inclusive security assessment. The consultant recommends that the client implement a 24x7 intrusion detection system and either maintain it consistently or outsource this work. This, combined with a solid network perimeter defense (i.e—stateful firewall) and host-based defenses (host-based IDS's, and other security measures), would be a solid foundation to insure the computer security of the customer.

The first thing that jumps out at me is the Watchlist traffic. For example, all the warnings entitled "Watchlist 000222 NET-NCFC" could be innocuous. However, traffic from these addresses are known to be from a computer network with known malicious activity. This could be malicious activity. In the interests of brevity I will only cite one of each type of violation where appropriate.

Lots of mail traffic going to different hosts on your network. Are these all mail servers?

```
         /SnortA6.txt:08/20-15:07:53.178909  [**] Watchlist 000222 NET-NCFC [**]
159.226.63.190:1172 -> MY.NET.253.43:25
        ./SnortA6.txt:08/20-15:09:15.767878  [**] Watchlist 000222 NET-NCFC [**]
159.226.63.190:1181 -> MY.NET.253.42:25
        ./SnortA6.txt:08/20-15:11:40.133450  [**] Watchlist 000222 NET-NCFC [**]
159.226.63.190:1227 -> MY.NET.253.41:25
        ./SnortA3.txt:08/17-00:16:01.965091  [**] Watchlist 000222 NET-NCFC [**]
159.226.5.222:2002 -> MY.NET.100.230:25
```

This same network is hitting your telnet server (many instances like this):

```
./SnortA4.txt:08/16-20:49:32.362606  [**] Watchlist 000222 NET-NCFC [**]
159.226.45.3:4628 -> MY.NET.6.7:23
```

There are also ftp-data sessions entering your network, so someone is downloading files from their network to yours (many instances like this):

```
        ./SnortA6.txt:08/20-15:12:54.957384  [**] Watchlist 000222 NET-NCFC [**]
159.226.114.129:21 -> MY.NET.162.199:1095
```

There also appears to be RPC traffic coming into your network from theirs. This makes me very wary as well, considering the many vulnerabilities of RPC:

```
        ./SnortA6.txt:08/20-15:12:56.375558  [**] Watchlist 000222 NET-NCFC [**]
159.226.114.129:37266 -> MY.NET.162.199:1096
```

There are many other traffic patterns coming from this domain into and out of yours. However, unless there is a compelling need for this type of traffic with your business, I would consider this hostile activity.

Other watchlist traffic includes the following:

```
        08/16-09:06:07.879318  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.61.247:2052 -> MY.NET.5.29:443
```

There are many items like thie one above in your logs. This represents a secure connection to one of your webservers. If you're not running a secure webserver at this address then this could definitely represent someone having compromised this box and is now trying to evade detection by running their communications over an encrypted connection.

There are many portscans coming in as well. Take, for example, 195.114.226.41. This host resolves to apollo-dh0040.multiweb.net. Is this a commonly visited webserver for your users? If not, you may have a problem here. It is not one of MultiWeb's listed domain name servers. Here's there whois listing:

```
[whois.internic.net]

Whois Server Version 1.3
```

```
Domain names in the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

   Domain Name: MULTIWEB.NET
   Registrar: NETWORK SOLUTIONS, INC.
   Whois Server: whois.networksolutions.com
   Referral URL: www.networksolutions.com
   Name Server: NS.NEDERNET.NL
   Name Server: DNS.MULTIWEB.NET
   Updated Date: 25-aug-2000


>>> Last update of whois database: Wed, 22 Nov 2000 21:15:13 EST <<<

The Registry database contains ONLY .COM, .NET, .ORG, .EDU domains and
Registrars.
```

There are literally hundreds (possibly thousands) of connections from this box to different machines on your network. Either everyone in your organization has this as their default homepage in their browsers or they've got every machine in your network mapped and portscanned.

You've also got traffic coming in bound for SUN-RPC ports on your hosts:

```
08/15-06:04:39.316658  [**] Attempted Sun RPC high port access [**]
205.188.179.33:4000 -> MY.NET.217.42:32771
```

This is a bad sign as well. If the target box here is a UNIX box, you may have trouble.

You've also got stealth portscans coming in :

```
08/15-10:24:40.165477  [**] spp_portscan: PORTSCAN DETECTED from 24.18.91.19
(STEALTH) [**]
```

This is almost certainly malicious activity and this user (cable-modem person judging by the 24. address) should be watched very carefully. Here is the rest of the activity from this user:

```
./SOOS8.txt:09/07-18:10:48.536410 24.18.91.19:1766 -> MY.NET.253.114:80
./SOOS17.txt:09/04-14:12:03.455691 MY.NET.222.110:0 -> 24.18.91.196:1318
./SOOS17.txt:09/04-14:12:42.827351 MY.NET.222.110:1318 -> 24.18.91.196:5190
./SOOS17.txt:09/04-14:17:14.474774 MY.NET.222.110:1318 -> 24.18.91.196:5190
./SOOS17.txt:09/04-15:57:58.935941 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-16:03:46.454139 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-16:05:35.194556 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-16:06:46.332850 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-16:09:52.106926 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-16:13:25.733016 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-16:15:21.867041 MY.NET.222.110:1356 -> 24.18.91.196:5190
```

```
./SOOS17.txt:09/04-16:42:57.967833 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-16:54:36.156095 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-17:03:19.597293 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-17:04:25.250546 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-17:05:49.027814 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-17:14:00.544065 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-17:15:52.837001 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-17:17:30.863235 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-17:20:15.583222 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-17:24:35.295292 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-17:26:12.550447 MY.NET.222.110:1356 -> 24.18.91.196:5190
./SOOS17.txt:09/04-17:27:05.086721 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-17:31:14.871789 MY.NET.222.110:0 -> 24.18.91.196:1356
./SOOS17.txt:09/04-17:34:05.064759 MY.NET.222.110:0 -> 24.18.91.196:1356
```

Data traveling from you're my.NET.222.110 port 0 to this user is extremely suspicious. It looks like this user has or is in the process of compromising this box. In fact, the stealth packet alone makes this user suspicious. In total, you've had 1885 STEALTH warnings and 163 RESERVEDBITS warnings from Snort. The possibility exists that this could be corrupted data coming from a bad network card. Here's a good example of this type of behavior:

```
./SnortS2.txt:Aug 17 00:04:51 24.23.198.174:2516 -> MY.NET.217.46:1386 XMAS
*1*F*P*U RESERVEDBITS
./SnortS2.txt:Aug 17 00:04:57 24.23.198.174:0 -> MY.NET.217.46:2516 FULLXMAS
*1SFRPAU RESERVEDBITS
./SnortS2.txt:Aug 17 00:07:03 24.23.198.174:2523 -> MY.NET.217.46:1396 NULL
********
./SnortS2.txt:Aug 17 00:33:00 24.23.198.174:0 -> MY.NET.217.46:2751 VECNA
21*****U RESERVEDBITS
./SnortS2.txt:Aug 17 00:34:52 24.23.198.174:2751 -> MY.NET.217.46:1431 NOACK
*1S*R**U RESERVEDBITS
./SnortS2.txt:Aug 17 00:40:15 24.23.198.174:0 -> MY.NET.217.46:2789 INVALIDACK
**S*RPAU
./SnortS2.txt:Aug 17 00:45:41 24.23.198.174:0 -> MY.NET.217.46:2855 NOACK
2**FR*** RESERVEDBITS
./SnortS2.txt:Aug 17 00:46:05 24.23.198.174:0 -> MY.NET.217.46:2855 INVALIDACK
21SFRPA* RESERVEDBITS
./SnortS2.txt:Aug 17 00:47:07 24.23.198.174:0 -> MY.NET.217.46:2897 INVALIDACK
2*SFR*AU RESERVEDBITS
./SnortS2.txt:Aug 17 00:54:10 24.23.198.174:0 -> MY.NET.217.46:2958 VECNA
2****P** RESERVEDBITS
./SnortS2.txt:Aug 17 00:56:36 24.23.198.174:2975 -> MY.NET.217.46:1472 NOACK
21**RP** RESERVEDBITS
./SnortS2.txt:Aug 17 00:57:41 24.23.198.174:0 -> MY.NET.217.46:2975 NOACK
21S****U RESERVEDBITS
./SnortS2.txt:Aug 17 00:59:38 24.23.198.174:2994 -> MY.NET.217.46:1480 VECNA
*1***P*U RESERVEDBITS
```

```
./SnortS2.txt:Aug 17 01:00:12 24.23.198.174:2994 -> MY.NET.217.46:1480 NOACK
2*S*RP*U RESERVEDBITS
./SnortS2.txt:Aug 17 01:05:16 24.23.198.174:0 -> MY.NET.217.46:3067 UNKNOWN
2**F*PA* RESERVEDBITS
./SnortS2.txt:Aug 17 01:09:13 24.23.198.174:3167 -> MY.NET.217.46:1501 NULL
********
./SnortS2.txt:Aug 17 01:09:48 24.23.198.174:3215 -> MY.NET.217.46:1505 NULL
********
./SnortS2.txt:Aug 17 01:10:51 24.23.198.174:0 -> MY.NET.217.46:3215 NOACK
2***RP*U RESERVEDBITS
./SnortS2.txt:Aug 17 01:11:52 24.23.198.174:3215 -> MY.NET.217.46:1505 NULL
********
./SnortS2.txt:Aug 17 01:12:33 24.23.198.174:0 -> MY.NET.217.46:3215 INVALIDACK
*1**RPAU RESERVEDBITS
./SnortS2.txt:Aug 17 01:13:13 24.23.198.174:3223 -> MY.NET.217.46:1515 SYNFIN
*1SF**** RESERVEDBITS
./SnortS2.txt:Aug 17 01:15:15 24.23.198.174:3223 -> MY.NET.217.46:1515
INVALIDACK **SF**AU
```

Most of the ports here are uncommon communications protocols with valid uses. They are not backdoors necessarily. The fact that this person is hitting inconsequential ports with many mangled packets suggests that these are false positives. However, here's a short excerpt of someone who is probably more malicious. This user had hundreds of portscans detected on many of your machines. Also, they were specifically targeting port 994, which is an encrypted IRC port (irc protocol over TLS SSL):

```
./SnortA3.txt:08/17-09:40:27.459430  [**] spp_portscan: PORTSCAN DETECTED from
130.149.41.70 (STEALTH) [**]
./SnortA3.txt:08/17-09:28:08.015858  [**] Null scan! [**] 130.149.41.70:1230 ->
MY.NET.217.46:994
./SnortA3.txt:08/17-11:50:26.287622  [**] Probable NMAP fingerprint attempt [**]
130.149.41.70:1157 -> MY.NET.217.46:994
```

The customer has also had hundreds of attempted connections to WinGate proxy server across many of their machines. The customer could not possibly have this many Socks proxies running on their network. Here is a short excerpt (relatively speaking):

```
./SnortA23.txt:09/13-11:52:55.939730  [**] WinGate 1080 Attempt [**]
216.179.0.37:4080 -> MY.NET.203.246:1080
./SnortA23.txt:09/13-11:58:59.752818  [**] WinGate 1080 Attempt [**]
207.114.4.46:2043 -> MY.NET.210.146:1080
./SnortA23.txt:09/13-12:15:44.656411  [**] WinGate 1080 Attempt [**]
63.238.214.65:4666 -> MY.NET.203.46:1080
./SnortA23.txt:09/13-12:17:41.552979  [**] WinGate 1080 Attempt [**]
213.167.203.238:1991 -> MY.NET.203.46:1080
./SnortA23.txt:09/13-12:29:28.682755  [**] WinGate 1080 Attempt [**]
213.167.203.238:2012 -> MY.NET.224.6:1080
./SnortA23.txt:09/13-12:47:20.685980  [**] WinGate 1080 Attempt [**]
```

```
216.67.82.110:1701 -> MY.NET.60.11:1080
./SnortA23.txt:09/13-13:06:33.850660  [**] WinGate 1080 Attempt [**]
64.86.5.250:1425 -> MY.NET.206.62:1080
./SnortA23.txt:09/13-13:28:06.952781  [**] WinGate 1080 Attempt [**]
216.179.0.37:1605 -> MY.NET.218.158:1080
./SnortA23.txt:09/13-14:24:44.284000  [**] WinGate 1080 Attempt [**]
64.86.6.250:2519 -> MY.NET.203.218:1080
./SnortA23.txt:09/13-14:49:21.355391  [**] WinGate 1080 Attempt [**]
206.204.30.83:20 -> MY.NET.97.165:1080
./SnortA23.txt:09/13-14:57:26.517702  [**] WinGate 1080 Attempt [**]
216.179.0.37:3297 -> MY.NET.202.214:1080
./SnortA23.txt:09/13-15:47:05.637232  [**] WinGate 1080 Attempt [**]
216.179.0.37:4571 -> MY.NET.203.246:1080
./SnortA23.txt:09/13-16:22:09.553313  [**] WinGate 1080 Attempt [**]
64.86.6.250:1073 -> MY.NET.225.202:1080
./SnortA23.txt:09/13-16:27:39.687686  [**] WinGate 1080 Attempt [**]
64.86.5.250:4862 -> MY.NET.205.114:1080
./SnortA23.txt:09/13-16:42:27.779876  [**] WinGate 1080 Attempt [**]
139.102.123.14:1590 -> MY.NET.221.106:1080
./SnortA23.txt:09/13-16:42:27.807696  [**] WinGate 1080 Attempt [**]
24.218.207.146:4293 -> MY.NET.221.106:1080
./SnortAle.txt:08/11-00:46:56.976538  [**] WinGate 1080 Attempt [**]
208.240.218.220:4390 -> MY.NET.217.46:1080
./SnortAle.txt:08/11-01:07:47.877025  [**] WinGate 1080 Attempt [**]
208.240.218.220:3384 -> MY.NET.217.46:1080
./SnortAle.txt:08/11-01:19:25.212249  [**] WinGate 1080 Attempt [**]
194.75.152.237:38892 -> MY.NET.98.185:1080
./SnortAle.txt:08/11-01:27:06.939036  [**] WinGate 1080 Attempt [**]
216.179.0.37:2940 -> MY.NET.60.8:1080
./SnortAle.txt:08/11-01:39:58.051582  [**] WinGate 1080 Attempt [**]
216.67.82.19:2743 -> MY.NET.98.138:1080
./SnortAle.txt:08/11-01:39:59.043073  [**] WinGate 1080 Attempt [**]
216.67.82.19:2743 -> MY.NET.98.138:1080
./SnortAle.txt:08/11-01:39:59.940287  [**] WinGate 1080 Attempt [**]
216.67.82.19:2743 -> MY.NET.98.138:1080
./SnortAle.txt:08/11-01:40:00.744956  [**] WinGate 1080 Attempt [**]
216.67.82.19:2743 -> MY.NET.98.138:1080
./SnortAle.txt:08/11-01:40:00.767608  [**] WinGate 1080 Attempt [**]
207.175.201.147:3096 -> MY.NET.98.138:1080
```

All of these different boxes attempting connections to this many different machines on your network tells me that your network address must have been published on an IRC somewhere. This is an inordinately large number of connection attempts to WinGate (969 total across all the logs I was presented with).

The customer has also had two instances where a virus was sent to their mail server. Once inside the network, Snort will not necessarily pick up the spread until it comes back out to the border of the customer's network (it is assumed that such a large network would be switched, and therefore the IDS would not see ALL traffic between ALL boxes. Here are the recorded virus alerts:

```
./SnortA4.txt:08/16-14:36:46.954418  [**] Happy 99 Virus [**] 128.8.198.101:12805 ->
MY.NET.6.35:25
./SnortA6.txt:08/20-15:41:12.157972  [**] Happy 99 Virus [**] 24.2.2.66:58102 ->
MY.NET.179.80:25
```

Installing antivirus software on all computers would be a recommendation to take care of this problem.

SNMP problems also exist in the customer's organization. The customer currently has a server set up with the SNMP community string set to public, and 6 machines have had access to that box. Here are listings:

```
./SnortA16.txt:09/07-18:45:07.737900  [**] SNMP public access [**]
MY.NET.98.190:1128 -> MY.NET.101.192:161
./SnortA7.txt:08/19-11:55:30.142169  [**] SNMP public access [**]
MY.NET.98.148:1117 -> MY.NET.101.192:161
./SnortA6.txt:08/20-19:52:20.993056  [**] SNMP public access [**]
MY.NET.98.191:1186 -> MY.NET.101.192:161
./SnortA6.txt:08/20-16:32:04.891367  [**] SNMP public access [**]
MY.NET.97.246:1085 -> MY.NET.101.192:161
./SnortA2.txt:08/15-20:15:48.753784  [**] SNMP public access [**]
MY.NET.97.154:1151 -> MY.NET.101.192:161
./SnortA3.txt:08/17-16:13:10.575484  [**] SNMP public access [**]
MY.NET.98.177:1047 -> MY.NET.101.192:161
```

This is not necessarily malicious activity, but it is definitely poorly configured. Leaving the default community string of PUBLIC is equivalent to leaving a blank administrator or root password on your machine.

The customer has also been scanned on numerous occasions for FTP servers across their network. Here is an example:

```
Aug 15 00:46:21 195.114.226.41:2416 -> MY.NET.1.174:21 SYN **S*****
Aug 15 00:46:21 195.114.226.41:2417 -> MY.NET.1.175:21 SYN **S*****
Aug 15 00:46:21 195.114.226.41:2418 -> MY.NET.1.176:21 SYN **S*****
Aug 15 00:46:21 195.114.226.41:2277 -> MY.NET.1.35:21 SYN **S*****
Aug 15 00:46:21 195.114.226.41:2274 -> MY.NET.1.32:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2276 -> MY.NET.1.34:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2353 -> MY.NET.1.111:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2350 -> MY.NET.1.108:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2278 -> MY.NET.1.36:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2280 -> MY.NET.1.38:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2281 -> MY.NET.1.39:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2279 -> MY.NET.1.37:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2419 -> MY.NET.1.177:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2420 -> MY.NET.1.178:21 SYN **S*****
Aug 15 00:46:22 195.114.226.41:2421 -> MY.NET.1.179:21 SYN **S*****
```

With someone this interested in finding an FTP server on the customer's network, they wi

In conclusion, this is not meant to be a complete risk and/or damage assessment. Judging

ANALYSIS PROCESS

In order to analyze the preceeding section I relied on a working knowledge of UNIX commands. The technique I employed was to look for key issues, and use the UNIX command-line interface to parse all the Snort logs we were given. For example, beginning with obvious keywords (STEALTH, WinGate, RESERVEDBITS, Virus, and any others I spotted while poring through the logs) I would go through the following steps:

First, I downloaded all the files used in the preceding section to a directory on my Linux box. Then I did the following:

```
# find ./ -name '*.txt' | xargs grep "keyword" [ | grep "keyword"] > "keyword"
```

This would find all the txt files in the directory, grep them for a certain keyword, grep again for another keyword if necessary, and then output the results to a file named by that keyword. Then, opening that file in another terminal window, I would look for suspicious listings, and if there were addresses that stood out in that list, I would repeat the step above for that address to get the rest of that user's activity. I found this process to be very effective in my analysis. I would have liked to use something like Snort-Snarf, but I didn't leave myself time to get that done.