Global Information Assurance Certification Paper

GIAC INTRUSION DETECTION CURRICULUM PRACTICAL ASSIGNMENT Version 2.7

Capitol SANS, Washington DC 2001
Toshi Iijima

Assignment 1- Network Detects

DETECT 1 -- Stimulus-Response

```
Nov  9 04:23:21 hostbe in.telnetd[1642]: refused connect from p14-tnt1.mel.ihug.com.au
Nov  9 04:24:34 hostca in.telnetd[25604]: connect from p14-tnt1.mel.ihug.com.au
Nov  9 04:25:01 hostca in.telnetd[25608]: connect from p14-tnt1.mel.ihug.com.au
Nov  9 04:36:42 hosty telnetd[193710]: refused connect from p14-tnt1.mel.ihug.com.au
Nov  9 04:37:04 hostmi in.telnetd[22068]: refused connect from p14-tnt1.mel.ihug.com.au
```

1. Source of trace:

This source was taken from http://www.sans.org/y2k/111000-1200.htm as submitted by Laura@edu

2. Detect was generated by:

These detects are the output from four different UNIX variants running TCP_WRAPPERS over the telnet service that has been sysloged to a central logging server. TCP_WRAPPERS is a program that makes it possible monitor incoming requests by restricting access to a system resource based on the connecting host's IP address and requested service. TCP_WRAPPERS is configured within the inetd.conf file so that when requests to services are spawned, they are intercepted by the tcpd daemon, which processes the request before handing it off to the requested service. The tcpd daemon is the actual program that logs the requests and evaluates the parameters of the incoming request against two configuration files – hosts.allow and hosts.deny. All accepted and refused connections are logged. It will not detect TCP half open connections. These configuration files are accessed in the order of hosts.allow and then hosts.deny, however only one of the files needs to exist. The format of the hosts.allow file is:

daemon_list: host

This host-based approach to security is still susceptible to IP spoofing unless TCP_WRAPPERS is compiled with –DPARANOID. This compile-time flag option forces tcpd to verify the client by performing a DNS reverse lookup, to ensure that the address to name and name to address are matched. If there is a mismatch in the DNS information the connection will be dropped.

The log output contains 7 fields that contain the following information about the TCP session:

1 - date, 2 - timestamp, 3 - dst host, 4 - daemon name, 5 - process ID, 6 - action taken, 7 - src host

3. Probability the source address was spoofed

It is most likely that the address was not spoofed, since the attacker would have nothing to gain had he not received the response generated by TCP stimulus.

The connection to the telnet service was recorded through the TCP_WRAPPERS, which was able to perform a reverse lookup on the source address and find the IP address of the host published in DNS, which further eliminates the possibility that the address was spoofed. Of course, this does not discount the probability that someone may have compromised access to the source hosts and has decided to make direct probes from these legitimate address sources.

4. Description of attack:

The attack shows a one to many connection attempts to telnet services on four different hosts. Three out of the four hosts refused the connection from p14-tnt1.mel.ihug.com.au, while hostca permitted the attacker to establish two separate telnet sessions as noted by the two separate telnet process IDs 25604 and 25608. This shows that telnet access is permitted for this host. The timestamp of the 5 recorded log entries are not within very tight time intervals, which suggests that this was a manual probe versus an automated probe through a script.

5. Attack mechanism:

This is the classical stimulus-response attack, attempting to directly contact telnet service on port 23. One can determine if the telnet service is listening or not listening for telnet requests, or if the host is being filtered through a wrapper such as TCP_WRAPPERS, set only to allow connections from trusted host addresses. If the telnet service is not being filtered, the attacker should be able to ascertain the OS of the host through the banner message elicited.

6. Correlation:

Since this information was a host-based intrusion detect log, it would be valuable to gather network based packet capture from log files from any packet-filtering device such as a firewall or IDS network sensor to correlate some events. A tcpdump or Snort output may show that prior to the host-based detect, there may have been stealth port scans performed to the network that these hosts reside on as a reconnaissance effort. The network-based detect logs most likely have captured anomalous TCP flag options being sent to the victims and half-open TCP connections which would not be detected by TCP_WRAPPERS which relies on a full TCP connect( ). It is also likely that other UDP or ICMP probes were performed against the victim's network, prior to the dates of the probes captured on the host's log files.

7. Evidence of active targeting:

From these detects alone it is not possible to conclude that this was an active attack. There are other stealth methods to determine if a service is listening on a port without being discovered, such as sending anomalous TCP flag options SYN-FIN, FIN, ACK, or Xmas scans using utilities like nmap. Also sending RESETS before a TCP connection is fully established (known as a half-open connection) are other methods of evasive maneuvers to perform reconnaissance. Most likely these stealth methods were already performed, but at this point the attacker is still determining the behavior of the telnet service available to see if it is being filtered through a host based ACL like TCP_WRAPPERS.

In future log analysis, it is very likely that there will be a great number of failed login attempts from the same host to host hostca in this example. This would indicate that the attacker is using brute-force tactic to crack a password for a login ID using a utility like "crak". When evidence of this nature is logged we can conclude that an active attack is being performed.

8. Severity:

Severity = (Critical + Lethal) – (System + Net Countermeasures) Severity
1= (4+5)-(3+5) possibly lethal if successfully establishes telnet connection.

9. Defensive recommendation:

The administrator of this host may want to consider placing these servers behind a packet-filtering device that blocks telnet access to their hosts, which seem to be exposed to the public. In addition the administrator may want to consider using ssh instead of telnet to provide remote administrative access. SSH can be compiled with the libwrap library from TCP_WRAPPERS to set filters on allowed source addresses. The sshd daemon configuration should also be set to deny root ssh access, prevent password logins, and only allow public key exchanges for authentication. Additionally a host based port monitoring tools such as Psionic's Port sentry should be installed so all TCP/UDP/ICMP connections to the box can be monitored. Psionic's tool has the ability to detect TCP half-open connections and other stealth scanning modes.

10. Multiple choice test questions based on trace:

Which one of these assumptions is most accurate?

a) A policy or ACL on a network-based packet-filtering device is selectively preventing some external hosts to connect to telnet services.
b) Telnet services for these hosts are listening on port numbers -- 1642, 25604, 25608, 193710, and 22068.
c) The attacker cannot perform OS fingerprinting to any of these hosts.
d) All the hosts will respond to telnet requests.

ANS: d We don't have any information regarding the firewall policy, so we can't make assumptions about statement (a)

DETECT 2 -- Crafted SYN-FIN Probe

```
[**] IDS198/SYN FIN Scan [**]
10/30-16:17:28.818059 216.103.84.187:9704 -> a.c.205.138:9704 TCP TTL:26 TOS:0x0 ID:39426
**SF**** Seq: 0x6D317CE6 Ack: 0x3FCCD487 Win: 0x404 00 00 00 00 00 00 .....
--
10/31-12:49:09.672439 216.103.84.187:9704 -> a.d.53.33:9704 TCP TTL:23 TOS:0x0 ID:39426
**SF**** Seq: 0x41D1E8D5 Ack: 0x118A5F74 Win: 0x404
10/31-12:49:09.701743 216.103.84.187:9704 -> a.d.53.35:9704  TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x41D1E8D5 Ack: 0x118A5F74 Win: 0x404
10/31-12:49:09.833783 216.103.84.187:9704 -> a.d.53.45:9704 TCP TTL:24 TOS:0x0 ID:39426
**SF**** Seq: 0x41D1E8D5 Ack: 0x118A5F74 Win: 0x404
```

1. Source of trace

The source of this trace was taken from GIAC http://www.sans.org/y2k/110800.htm

2. Detect was generated by:

These log detects are alerts that were generated by Snort using a rule-set that falls under the category of "scans". The following rule-set detects SYN/FIN scans. The rule can be translated as "Set an alert for all TCP based packets with the TCP flags SYN/FIN set on from any source not belonging to my network to destination my network".

alert tcp !$HOME_NET any -> $HOME_NET any (msg:"IDS198 - SCAN-SYN FIN";flags:SF;)

However, to differentiate this particular type of SYN-FIN scan from other scans it may be a prudent idea to build a more specific IDS signature. Adding too much items in the rule could create a false negative situation– no detection.

alert tcp !$HOME_NET any -> $HOME_NET any (msg:"SF with ID 39426";flags:SF;ID:39426)

3. Probability the source address was spoofed:

At first sight it appears that there is high probability that the address was not spoofed. The TCP ttl values fall relatively around 23-26 so it seems that the address is coming from a fixed location. However, the attacker that is logged in this detect might not be the IP address of the real attacker, but instead it might be used by the real attacker as a go between to perform these scans.

4. Description of attack:

The detect raises a lot of suspicion since every TCP/IP packet variable remains the same across each detect. The trace shows that the source address remains constant, the source port and destination port are equal. The TCP ID number also remains fixed at 39426, the sequence number and ACK number is fixed, and the window size is always 1028 (0x404).

These packets were detected because it appeared as a standard SYN-FIN probe. Upon closer examination it appears like the attacker is probing for some rpc.statd based back doors installed on TCP port 9704.

5. Attack mechanism:

Initially this looks like a typical reconnaissance scan. However, this may not be your typical reconnaissance scan. Teri Bidell has investigated the possibility that this may be a spoofed reconnaissance attack, known as "idle-scanning". This form of spoofed scanning takes advantage of predictable methods in which IP IDs are incremented

by many operating systems. This attack requires that the attacker has access to a host whose operating system has a predictable way of incrementing the IP ID and that the host is relatively inactive or "idle" so the attacker can measure the incremental value of the IP ID each time they send a packet through this spoofed host. Hence, the name of this attack is referred to as "idle-scanning".

For this attack to work the attacker makes some assumptions about whether a port is open or closed by measuring how much the IP ID of the "idle host" has been incremented. If a port is opened on the victim, it should respond to the "spoofed" SYN request with a SYN-ACK back to the "idle host". Since, the "idle host" did not send the SYN request it will respond to the SYN-ACK with a RST and consequently increment the IP ID on the "idle host" by more than 1. If the port is closed a RST will be sent back to the "idle host" and thus the IP ID will only be incremented by 1 since the "idle host" will not reply back to the RST. After sending the spoofed reconnaissance scans using the IP address of the idle host, the attacker determines the response by the victim to the spoofed host by checking how the IP Ids on the "idle host" were incremented using a tool like "hping". Due to the nature of the attack it can be used in distributed spoofed scan and is a relatively slow attack process.

6. Correlation:

- http://www.securityfocus.com/frames/?focus=ids&content=/focus/ids/articles/icmptools.html
- http://www.sys-security.com/archive/securityfocus/icmptools.html
- Possible attack using "Idlescan" as described by Teri Bidwell in his practical exam posted at http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc
- Link to hping2 by Antirez, http://www.kyuzz.org/antirez/hping/
- As describe in CVE-2000-0666 at http://cve.mitre.org/
- From advisory posted by CERT at http://www.cert.org/advisories/CA-2000-17.html
- Also from postings at SANS found on URL http://www.sans.org/y2k/110600.htm and http://www.sans.org/y2k/110100-1230.htm

7. Evidence of active targeting:

NO. The attacker is trolling for hosts vulnerable to the rpc.statd buffer overflow at this stage.

8. Severity:

(Critical + Lethal) – (System + Net Countermeasures) = Severity
(2+2)-(4+4)=2

9. Defensive recommendation:

Upgrade the version of rpc.statd if the operating system is Linux and disable the rpc.statd service if it's not really required. Debian, Red Hat and Connectiva have all released advisories on this matter. Additionally, this service should be blocked from external organizations by setting a policy in the firewall.

10. Multiple choice test question:

Which is the most accurate statement?

a) The packet has been crafted
b) This type of packet will be detected by the Snort pre-processor plugin
c) alert tcp !$HOME_NET any -> $HOME_NET any (flags: SF;msg: "SYN FIN Scan";ID:39426; win: 0x404)
d) None of the above

ANS: a, Clearly all the fields within the packets are crafted, choice (c) the syntax for window size should be dsize. Snort Pre-Processor functions don't deal directly with filtering packet contents.

DETECT 3 -- DOS attempt through Buffer overflow of Telnet Login ID

From Solaris snoop packet output using command `snoop –i <capture file>`

```
21  0.04718   victim -> attacker   TELNET R port=1038
22  0.00040   attacker-> victim    TELNET C port=1038
23  0.00018   attacker -> victim   TELNET C port=1038
24  0.00009   victim -> attacker   TELNET R port=1038
25  0.00045   attacker -> victim   TELNET C port=1038
26  0.00009   victim -> attacker   TELNET R port=1038
27  0.00720   attacker -> victim   TELNET C port=1038
28  0.00025   victim -> attacker   TELNET R port=1038
29  0.00046   attacker -> victim   TELNET C port=1038
30  0.00156   victim -> attacker   TELNET R port=1038
31  0.00042   attacker -> victim   TELNET C port=1038
32  0.02442   attacker -> victim   TELNET C port=1038 UUUUUUUUUUUUUUUUUUUU
33  0.00081   attacker -> victim   TELNET C port=1038 UUUUUUUUUUUUUUUUUUUU
34  0.00015   attacker -> victim   TELNET C port=1038
35  0.00031   attacker -> victim   TELNET C port=1038
```

From Solaris snoop packet display using –vx options

Only packet number 32 and 33 need examination since they seem to contain some strange data.

```
31  0.00042   attacker -> victim      TELNET C port=1038

       0: 0800 2083 465f 0060 978b ac22 0800 4500    .. .F_.`..."..E.
      16: 0040 01e4 4000 4006 2006 0a03 02c7 0a03    .@..@.@. .......
      32: 0202 040e 0017 adf8 65a3 0534 f906 8018    .......øe..4....
      48: 7d78 519f 0000 0101 080a 0009 fcf8 0080    }xQ.........ø..
      64: 3b04 fffc 1fff fc23 fffc 27ff fc24         ;......#..'..$

32  0.02442 attacker -> victim        TELNET C port=1038 UUUUUUUUUUUUUUUUUUUU

       0: 0800 2083 465f 0060 978b ac22 0800 4500    .. .F_.`..."..E.
      16: 05dc 01e5 4000 4006 1a69 0a03 02c7 0a03    ....@.@..i......
      32: 0202 040e 0017 adf8 65af 0534 f906 8018    .......øe..4....
      48: 7d78 36e1 0000 0101 080a 0009 fcfa 0080    }x6.............
      64: 3b04 5555 5555 5555 5555 5555 5555 5555    ;.UUUUUUUUUUUUUU

                                  // --- truncated to fit page --- //

    1472: 5555 5555 5555 5555 5555 5555 5555 5555    UUUUUUUUUUUUUUUU
    1488: 5555 5555 5555 5555 5555 5555 5555 5555    UUUUUUUUUUUUUUUU
    1504: 5555 5555 5555 5555 5555                   UUUUUUUUUU

33  0.00081 attacker -> victim        TELNET C port=1038 UUUUUUUUUUUUUUUUUUUU

       0: 0800 2083 465f 0060 978b ac22 0800 4500    .. .F_.`..."..E.
      16: 0482 01e6 4000 4006 1bc2 0a03 02c7 0a03    ....@.@.........
      32: 0202 040e 0017 adf8 6b57 0534 f906 8018    .......økW.4....
      48: 7d78 dd3d 0000 0101 080a 0009 fcfa 0080    }x.=............
      64: 3b04 5555 5555 5555 5555 5555 5555 5555    ;.UUUUUUUUUUUUUU
      80: 5555 5555 5555 5555 5555 5555 5555 5555    UUUUUUUUUUUUUUUU

                              // --- truncated to fit page ---  //
```

```
1120: 5555 5555 5555 5555 5555 5555 5555 5555    UUUUUUUUUUUUUUUU
1136: 5555 5555 5555 5555 5555 5555 5555 5555    UUUUUUUUUUUUUUUU
1152: 5555 5555 5555 5555 5555 5555 5555 5555    UUUUUUUUUUUUUUUU
```

1. Source of trace

Packets captured from our company's network with address mask of 16bit. The information has been obfuscated using the host names "attacker" and "victim" that correspond to source and destination respectively.

2. Detect was generated by:

The tool used to capture these packets is Snoop which comes by default with Sun Microsystems Solaris OS. Snoop performs packet capturing and displays the contents of saved binary capture files. It works similarly to a BSD based counterpart utility called "tcpdump". The output above was displayed from a previously saved binary capture file with the following command:

Solaris # snoop –i <capture file>

35	0.00031	Attacker	victim	TELNET	C port=1038
Relative sequence number –	relative timestamp –	source –	destination	-- service –	port number

The second output was generated using this flag option -xv, which returns a formatted output. The second output shows the same information above and in addition shows the details of the data portion of the payload in hexadecimal and ASCII translation.

3. Probability the source address was spoofed

No, this attack requires that the attacker have a full TCP connection with the remote host so it can send a large string as the user ID during the interactive telnet login prompt.

4. Description of attack:

The attack falls into the category of denial of service (DOS). The attacker has sent a very large string of ASCII character 'U's as the user ID in an attempt to overflow the login prompt.

5. Attack mechanism:

A remote user can cause some telnet servers to stop responding to requests or consume a great amount of CPU processing time by sending a stream of binary zeros to the telnet server. This can easily be reproduced from any system using the utility called "netcat" with an input of /dev/zero or any large file, with a command such as "nc target.host 23 < /dev/zero". The Windows 2000 Telnet Server stops responding to requests after a few seconds. If the telnet server is set to restart upon failure, it will restart and immediately fail. This will occur repeatedly until the telnet server exceeds its restart count, at which point the service remains down.

6. Correlation:

The host "victim" locally logged telnet connections from attacker through syslog facility by running the inetd daemon with the extra flag option –t turned on. By default Solaris servers start the inetd daemon with just the –s option. The additional –t option allows the inetd daemon to make reverse lookups on the connecting host.

From Solaris log file with inetd running with –st options

```
Nov 22 10:59:10 victim inetd[2964]: telnet[3103] from attacker 1038
Nov 22 10:59:10 victim inetd[2964]: telnet[3106] from attacker 1039
Nov 22 11:09:21 victim inetd[2964]: telnet[3153] from attacker 1040
Nov 22 11:15:06 victim inetd[2964]: telnet[3171] from attacker 1105
Nov 22 11:15:10 victim inetd[2964]: telnet[3173] from attacker 1121
Nov 22 11:15:29 victim inetd[2964]: telnet[3175] from attacker 1138
Nov 22 11:15:32 victim inetd[2964]: telnet[3178] from attacker 1157
Nov 22 11:15:44 victim inetd[2964]: telnet[3181] from attacker 1177
Nov 22 11:16:09 victim inetd[2964]: telnet[3184] from attacker 1204
Nov 22 11:16:26 victim inetd[2964]: telnet[3187] from attacker 1211
Nov 22 11:16:56 victim inetd[2964]: telnet[3190] from attacker 1213
```

- As describe CAN-2000-0480 at http://cve.mitre.org/ (Shadow Op Dragon Server Multiple DoS Vulnerabilities)
- SecureXpert Labs Advisory [SX-20000620-1] - Denial of Service vulnerability in Microsoft Windows 2000 Telnet Server

7. Evidence of active targeting:

Yes. The attacker is already aware that port 23 for telnet service was available and decided to see if they could overflow the buffer at the login prompt.

8. Severity:

(Critical + Lethal) – (System + Net Countermeasures) = Severity
(2+2)-(4+4)=1

9. Defensive recommendation:

Close telnet services or restrict access to this service using some ACL feature either on the firewall on the host using something like TCP_WRAPPERS.

10. Multiple choice test question:

This type of attack tries to take advantage of what vulnerability?

a) Telnet access on port 1038
b) Sending packets at a super high rate to the victim
c) Memory stack overflow
d) Attempt to login using a password that matches the string of the userID

ANS: c

DETECT 4 – Spoofed RSTs

Legend:

Lines of special interest
TCP RST flag
victim (10.3.2.44) at 00:A0:CC:DA:4D:B3
gateway (10.3.2.1) at 00:40:05:36:E8:42
attacker (10.3.2.6) at 00:40:05:40:AD:22

View 1 -- TCPDUMP read of saved binary capture file with
(Line numbers not part of tcpdump output on left for convenience)

1. 01:47:34.700065 eth0 P victim.1166 > tmtu.mt.rs.els-gms.att.net.domain: 42+ A? www.netscape.com. (34) (ttl 128, id 45650)
2. 01:47:34.751018 eth0 P tmtu.mt.rs.els-gms.att.net.domain > victim.1166: 42 5/2/2 www.netscape.com. CNAME www-mv.netscape.com., www-mv.netscape.com. A home-v2.websys.aol.com, www-mv.netscape.com. A home-v3.websys.aol.com, www-mv.netscape.com. (198) (DF) (ttl 243, id 13699)
3. 01:47:34.752602 eth0 P victim.1167 > home-v2.websys.aol.com.www: S 3617610196:3617610196(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id 45651)
4. 01:47:34.854484 eth0 P home-v2.websys.aol.com.www > victim.1167: S 3137704541:3137704541(0) ack 3617610197 win 33580 <nop,nop,sackOK,mss 1460> (DF) (ttl 234, id 46829)
5. 01:47:34.854708 eth0 > victim.1167 > home-v2.websys.aol.com.www: R 3617610197:3617610197(0) win 0 [tos 0x10] (ttl 48, id 37919)
6. 01:47:34.854680 eth0 P victim.1167 > home-v2.websys.aol.com.www: . 1:1(0) ack 1 win 17520 (DF) (ttl 128, id 45653)
7. 01:47:34.854921 eth0 > home-v2.websys.aol.com.www > victim.1167: R 3137704542:3137704542(0) win 0 [tos 0x10] (ttl 48, id 14254)
8. 01:47:34.855448 eth0 P victim.1167 > home-v2.websys.aol.com.www: P 1:434(433) ack 1 win 17520 (DF) (ttl 128, id 45654)
9. 01:47:34.855632 eth0 > home-v2.websys.aol.com.www > victim.1167: R 3137704542:3137704542(0) win 0 [tos 0x10] (ttl 48, id 39235)
10. 01:47:34.994154 eth0 P home-v2.websys.aol.com.www > victim.1167: R 3137704542:3137704542(0) win 0 (DF) (ttl 91, id 46830)

View 2 -- TCPDUMP read of saved binary file with –e option for ether-link level output and -v
(Line numbers not part of tcpdump output on left for convenience)

1. 01:47:34.700065 eth0 P **0:a0:cc:da:4d:b3 0:40:5:36:e8:42** ip 76: victim.1166 > tmtu.mt.rs.els-gms.att.net.domain: 42+ A? www.netscape.com. (34) (ttl 128, id 45650)
2. 01:47:34.751018 eth0 P **0:40:5:36:e8:42 0:a0:cc:da:4d:b3** ip 240: tmtu.mt.rs.els-gms.att.net.domain > victim.1166: 42 5/2/2 www.netscape.com. CNAME www-mv.netscape.com., www-mv.netscape.com. A home-v2.websys.aol.com, www-mv.netscape.com. A home-v3.websys.aol.com, www-mv.netscape.com. (198) (DF) (ttl 243, id 13699)
3. 01:47:34.752602 eth0 P **0:a0:cc:da:4d:b3 0:40:5:36:e8:42** ip 62: victim.1167 > home-v2.websys.aol.com.www: S 3617610196:3617610196(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id 45651)
4. 01:47:34.854484 eth0 P **0:40:5:36:e8:42 0:a0:cc:da:4d:b3** ip 62: home-v2.websyss.aol.com.www > victim.1167: S 3137704541:3137704541(0) ack 3617610197 win 33580 <nop,nop,sackOK,mss 1460> (DF) (ttl 234, id 46829)
5. 01:47:34.854708 eth0 > 0:40:5:40:ad:22 **0:40:5:36:e8:42** ip 54: victim.1167 > home-v2.websys.aol.com.www: R 3617610197:3617610197(0) win 0 [tos 0x10] (ttl 48, id 37919)
6. 01:47:34.854680 eth0 P **0:a0:cc:da:4d:b3** 0:0:0:0:0:1 ip 60: victim.1167 > home-v2.websys.aol.com.www: . 1:1(0) ack 1 win 17520 (DF) (ttl 128, id 45653)
7. 01:47:34.854921 eth0 > 0:40:5:40:ad:22 **0:a0:cc:da:4d:b3** ip 54: home-v2.websys.aol.com.www > victim.1167: R 3137704542:3137704542(0) win 0 [tos 0x10] (ttl 48, id 14254)
8. 01:47:34.855448 eth0 P **0:a0:cc:da:4d:b3 0:40:5:36:e8:42** ip 487: victim.1167 > home-v2.websys.aol.com.www: P 1:434(433) ack 1 win 17520 (DF) (ttl 128, id 45654)
9. 01:47:34.855632 eth0 > 0:40:5:40:ad:22**0:a0:cc:da:4d:b3** ip 54: home-v2.websys.aol.com.www > victim.1167: R 3137704542:3137704542(0) win 0 [tos 0x10] (ttl 48, id 39235)
10. 01:47:34.994154 eth0 P 0:40:5:40:ad:22 **0:a0:cc:da:4d:b3** ip 60: home-v2.websys.aol.com.www > victim.1167: R 3137704542:3137704542(0) win 0 (DF) (ttl 91, id 46830)

1. Source of trace

This detect was generated in a lab environment. The test environment consisted of a firewall with one interface connecting to the public Internet space and the other interface was connected to the local hub. The private LAN has been assigned a network address of 10.3.2.0/8 and has only three addresses on this network including one of the firewall interfaces. On the internal network there are two hosts, the victim is a Windows 98 desktop and the other host is the attacker, which is a Linux system, kernel version 2.3. The firewall policy is set to allow any internal host to send any protocol to any destination. The tcpdump capture is performed on the gateway, which is running OpenBSD 2.7.

2. Detect was generated by:

The session was logged via tcpdump to a binary file. Saved binary files can be reviewed later using BFP filter syntax to extract more precise information that provide different vantage points to view the session. In the first view (detect -- view 1) the output is generated using tcpdump as follows:

tcpdump –v –r capture_file

The second view (detect -- view 2) provides information at the Ethernet layer so we can see the actual MAC associated with the IP. This is outputted with tcpdump with the –e flag. The output of the packet capture file was line-numbered to make it easy to reference the information.

3. Probability the source address was spoofed

From looking at the first view 1 of the tcpdump output it doesn't look like there is any spoofing going. What we see seems to only be some transactions that minimally involve 3 hosts. In line 1 we see that the host victim has requested a DNS lookup up for the A record of www.netscape.com, which is being performed by the primary DNS server used by the test LAN. Line 2 shows that 5 records have been found, 2 authoritative servers, and 2 additional records were found. Line 3 and 4 shows the TCP handshake begin for this Web request. In lines 5 and 7 both ends of the conversation are issuing a RST flag and immediately terminate the session. From this view we might immediately

conclude that something has gone wrong at the application level and might pass this off as a problem with the users desktop environment. In other cases, if the network were heavily loaded, we might look for retransmissions or collision on the routing devices.

After examining View 2 we can conclude that the address was spoofed. View 2 shows somewhat of a different picture. It seems that the MAC address of the packets responsible for setting the RSTS are not being sent by the MAC address that should be associated with the IP address of the hosts in this conversation. Since the remote Web site is on a different broadcast segment than the local network where the packets were examined, the MAC is associated with the MAC of the last routing device on the network, which is the gateway. In fact we see that the attackers MAC address seems to be associated with the IP address of both ends of the session.

victim (10.3.2.44) at 00:A0:CC:DA:4D:B3
gateway (10.3.2.1) at 00:40:05:36:E8:42
attacker (10.3.2.6) at 00:40:05:40:AD:22

4. Description of attack:

The TCP connections seem to be sending RSTs for each packet that has the ACK bit set. When viewing the packets in a tcpdump file, at first glance one can't but help notice that there is one RST for every SYN. At first it looks very similar to the signature in a half-open attack, where the attacker tears down the 3-way handshake before it can be completed in order to do some sly port scanning. However, when you look more closely at direction of RSTs being delivered, it becomes apparent that the RSTs are being delivered from both ends. Usually in a TCP half-open attack, all the RSTs are all generated by one host that is initiator of the session, which is the attacker. When one of the hosts in communication sets RSTs this is normal, since it is usually some indication that there is a problem at the application layer, or some other problems that force the application to abort.

5. Attack mechanism:

"Rstfuck" is run at the command line only requires three arguments to execute.

Myshell # ./rstfuck
usage: rstfuck <interface> <ip> <port>

The "rstfuck" utility provides a window so you can see the details of the connections being reset in real time.

./rstfuck: filter is "tcp and host 10.3.2.44 and port 80"
192.88.209.22:80 -> 10.3.2.44:1158 [ACK 3563863614] RST!
10.3.2.44:1158 -> 192.88.209.22:80 [ACK 632650009] RST!
167.216.133.33:80 -> 10.3.2.44:1159 [ACK 3566227035] RST!
10.3.2.44:1159 -> 167.216.133.33:80 [ACK 47354179] RST!
205.164.217.39:80 -> 10.3.2.44:1161 [ACK 3568560282] RST!
10.3.2.44:1161 -> 205.164.217.39:80 [ACK 148145322] RST!
216.92.92.54:80 -> 10.3.2.44:1163 [ACK 3592519675] RST!
10.3.2.44:1163 -> 216.92.92.54:80 [ACK 2439731128] RST!
10.3.2.44:1163 -> 216.92.92.54:80 [ACK 2439731128] RST!
216.15.51.195:80 -> 10.3.2.44:1165 [ACK 3597645648] RST!
10.3.2.44:1165 -> 216.15.51.195:80 [ACK 2807311394] RST!
207.200.89.225:80 -> 10.3.2.44:1167 [ACK 3617610197] RST!
10.3.2.44:1167 -> 207.200.89.225:80 [ACK 3137704542] RST!
10.3.2.44:1167 -> 207.200.89.225:80 [ACK 3137704542] RST!
204.71.202.160:80 -> 10.3.2.44:1169 [ACK 3620098431] RST!
10.3.2.44:1169 -> 204.71.202.160:80 [ACK 1700196723] RST!
10.3.2.44:1169 -> 204.71.202.160:80 [ACK 1700196723] RST!
66.38.151.10:80 -> 10.3.2.44:1171 [ACK 3626206221] RST!
10.3.2.44:1171 -> 66.38.151.10:80 [ACK 2473585739] RST!
10.3.2.44:1171 -> 66.38.151.10:80 [ACK 2473585739] RST!
207.46.209.243:80 -> 10.3.2.44:1173 [ACK 3719194781] RST!
10.3.2.44:1173 -> 207.46.209.243:80 [ACK 535204514] RST!
10.3.2.44:1173 -> 207.46.209.243:80 [ACK 535204514] RST!

Basic caveats for using rstfuck at the moment:

- Since the application sniffs packets off the network, the attacker needs to on the same network segment where it is possible to view the TCP packets being sent across the network in order to be effective at all.
- This version of rstfuck does not spoof the MAC address
- Can only target single hosts and single port service, but can be rewritten to target network classes and port ranges.
- Rstfuck can also be compiled to send RSTs when it sees a packet with the SYN bit set.

The source code can be compiled on Linux, but it requires the libpcap and libnet library packages. The first library set is used so that the utility can directly capture network packets much in the same way that tcpdump allows packets to be captured. Using the Libnet library, it is very easily to craft packets at a very low level.

6. Correlations:

There are many applications and devices on the network today that already rely on the principle of terminating unwanted sessions tearing down TCP sessions through sending RSTS. For example there are some firewall brands that do this to prevent attackers from probing. There are other devices that do this to prevent an exorbitantly large number of lingering incomplete TCP handshake sessions from causing DOS, and recently there are URL filtering devices that will reset Web requests for unauthorized content. Snort is just another example of an application that has a feature called "Flexible Responses on Hostile Connection Attempts" that relies on sending RSTs based on detection criteria set by the security administrator. Snort can be compiled with the Libnet library which can be obtained from the URL http://www.packetfactory.net/Projects/Libnet/ . Libnet is a collection of routines to help with the construction and handling of network packets and it provides a portable framework for low-level network packet shaping, handling and injection. Libnet features portable packet creation interfaces at the IP layer and link layer, as well as a host of supplementary and complementary functionality. Sending RSTs against potentially hostile hosts created these applications, however, RSTs can be harmful especially when an attacker is spoofing the RSTs.

Normally the reset option is used by TCP to abruptly terminate a user session at the application layer.

- Sample source code rstfuck.c was obtained from distribution of Libnet-1.0.1b from http://www.packetfactory.net/Projects/Libnet/

Requires libpcap, the Packet Capture library which can be acrid from http://ee.lbl.gov/

*** The author of the program does not bear any responsibility for the misuse of the program, but has written this as a demonstration of some inherent protocol weaknesses, it is for educational purposes only.

7. Evidence of active targeting:

Yes, the attacker wants to prevent the user from being able to communicate with the remote service.

8. Severity:

(Critical + Lethal) – (System + Net Countermeasures) = Severity
(2+2)-(4+4)=1

The severity is very critical since this would prevent any TCP session. Within a brokerage firm it could potential break and or stop trading.

9. Defensive recommendation:

It is not possible for Snort to detect these types of attacks currently, since there are no features supporting tracking IDS signatures at the MAC layer.

The defensive mechanism would need to track the following characteristic of the attack:

- RSTS sent by both source and destination for every ACK or SYN sent by remote.
- Hard to detect the attack -- though it is automated the ACK and SYNs are immediately followed by RSTS, because pattern intervals are triggered in response to the pace of human execution times.
- MAC address for source and destination seem to be the same.

10. Multiple choice test question:

Which is true?

a) The attack is preventing all TCP and UDP traffic.
b) The MAC address is eth0
c) The RSTs seem to be sent by the requestor and the remote site
d) The MAC address is causing the session to terminate

ANS: c

Assignment 3 - "Analyze This" Scenario

The following analysis was performed from intrusion detects caught by the IDS tool Snort. The data collected was 53MG in size and comprised of 53 days worth of data, however the data is not complete due to power failures and or problems with log rotation. The log files are provided in three plain-text formats. The first set of data was based on the output of "Snort Alert Reports", the second format is the output form "Snort Scan Reports", and the third format contains console output, the result of running snort with flag option –v. The network under attack has chosen to obfuscate the identity of their network by referring to their network as MY.NET, a B class network.

The data from the "Snort Alert Reports" shows that external and internal hosts contacted a total of 35,846 hosts. The attackers were able to reach 254 subnets within MY.NET and 46% of these subnets received full network scans to 254 hosts plus the broadcast address 0 and 255.

Overview of Attack

Total number of alerts detected	110,534
Number of attack signatures detected	20
Total number of IP addresses in MY.NETscanned	35,846
Number of subnets accessed by intruders	254
Number of subnets receiving full scans to 254 hosts and to broadcast address 0 and 255	117 (46% of network environment)

List of detected Attacks and Probes by Volume

From Sept 26 – Nov 22 2000 (53 days)

Attack Signature	Detects	Percentage
SYN-FIN scan!	56,250	37.24
Portscan (includes TCP/UDP and stealth modes)	40,504	26.82
Watchlist 000220 IL-ISDNNET-990517	30,998	20.52
Watchlist 000222 NET-NCFC	8,166	5.41
WinGate 1080 Attempt	4,802	3.18
TCP SMTP Source Port traffic	2,893	1.92
Attempted Sun RPC high port access	2,542	1.68
Broadcast Ping to subnet 70	1,813	1.20
Back Orifice	1,697	1.12
SNMP public access	468	0.31
Null scan!	283	0.19
SMB Name Wildcard	218	0.14
Queso fingerprint	142	0.09
NMAP TCP ping!	96	0.06
SUNRPC high-port access!	60	0.04
Connect to 515 from inside	56	0.04
Probable NMAP fingerprint attempt	15	0.01
External RPC call	13	0.01
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	13	0.01
Tiny Fragments - Possible Hostile Activity	7	0.004
Happy 99 Virus	2	0.001

Making Sense of Attacks Committed

To understand the level of intrusion into MY.NET, we will need to identify the hacker's techniques and their intentions behind the attacks that caused alarms to trigger. To accomplish an attack there are certain stages an attacker must complete. Before an attack is considered, information is first gathered so the hacker can gain enough understanding about the network topology, learn more about the reachable targets and the vulnerabilities of the operating systems and or applications that will enable a point of attack. At this stage, the attacker will employ techniques that fall into the category of "reconnaissance". The intent of these techniques is discovery. Network mapping can be used to find different subnets of a network, while stealth scanning methods can be used to identity devices on the network. To further their awareness, hackers use stimulus-response techniques to understand the behavior of their target's OS and the specifics of the applications. The second stage is the actual exploitation of a target, known as "active targeting". In this stage if there is appropriate levels of logging through IDS sensors and system level logging it is possible to see evidence that shows the hacker targeting common vulnerabilities in applications or OS weaknesses. In the later stages of intrusion, the attackers tries cover their tracks in the initial exploit and install backdoor tools to allow continued access as well as for further garnering of information to expand their attack efforts.

Reconnaissance Efforts

Reconnaissance tactics make up for the majority of detected incidents. Due to their nature, it is not unusual to see such high volume. Usually this method is a shotgun approach to see how many random hosts it can contact in a short period of time. Once the information has been gathered it is used to distill a more targeted approach by understanding what kind of protocols, ports, and OS systems are visible. The severity of the incident in itself is very low, but in terms of the information that it provides, reconnaissance probes can end up being deadly instances which leak information about one's network and computing environment.

SYN-FIN Scan
37% of the alerts detected were packets with the SYN-FYN flags set (Data from OO files were not included in this stat). The attack begins by sending packets with the SYN/FIN scan which sets the flag options in an attempt to increase the possibility of evasively by-passing the firewall and reaching the intended victims. It seem that the packet filtering device or firewall protecting this network does not block SYN-FIN flagged packets. What makes these probes even more suspicious is the fact that both the source and destination ports are equal and that the source port is assigned to a low port of some well known services (WKS).

Most of the SYN-FIN scans were trolling for 21(FTP), 53(DNS), 9704(rpc.statd backdoor), and 27374 (subseven). The chart below shows that half of the SYN-FIN probes were scouting for FTP servers. They were most likely looking for FTP buffer overflow exploits.

Breakdown of SYN-FIN Alerts

Ports Probed	Alerts Generated
21	37,354
53	22,812
9704	14,200
27374	3,572

After scanning the network with SYN-FIN probes, there were 3 IP addresses that actively targeted hosts in an attempt to perform RPC and SMTP source port attacks.

Intruders	Type of active targeting performed after SYN-FIN probe
210.101.101.110	RPC attack
211.46.110.81	External RPC call, Sun RPC hi-port, SMTP source port
24.7.227.215	External RPC call, SMTP source port

Outbound Activity from MY.NET
Starts: 08/17/00 -- Ends: 11/23/00

- Lots of nntp news action to reader4.news.rcn.net (207.172.3.46) 119/tcp

- Chat/ICQ, NetMeeting, AOL/Netscape Instant Messenger (Port 5190)

Inbound Probes detected (from OO files)
Starts: 08/17/00 -- Ends: 11/23/00

Summary of TCP Flag Combinations

Number of unique attack sources:	210
Number of unique hosts probed:	26,438
Number of subnets probed:	149
Total number of packets received:	63,398
Total number TCP option combos:	67
SYN-FYN combo	98%

Total of 62,591 reports of TCP ID set to 39426

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-20:54:40.721682 130.89.229.48:53 -> MY.NET.1.58:53
TCP TTL:32 TOS:0x0 ID:39426
**SF**** Seq: 0x1F4217D2   Ack: 0xD6B8CBF   Win: 0x404
00 00 00 00 00 00                          ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-20:54:40.742690 130.89.229.48:53 -> MY.NET.1.59:53
TCP TTL:32 TOS:0x0 ID:39426
**SF**** Seq: 0x4C9CDBF1   Ack: 0x41B2195E   Win: 0x404
00 00 00 00 00 00                          ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-20:54:40.763462 130.89.229.48:53 -> MY.NET.1.60:53
TCP TTL:32 TOS:0x0 ID:39426
**SF**** Seq: 0x4C9CDBF1   Ack: 0x41B2195E   Win: 0x404
00 00 00 00 00 00                          ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-20:54:40.824459 130.89.229.48:53 -> MY.NET.1.63:53
TCP TTL:32 TOS:0x0 ID:39426
**SF**** Seq: 0x4C9CDBF1   Ack: 0x41B2195E   Win: 0x404
00 00 00 00 00 00                          ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-20:54:40.863524 130.89.229.48:53 -> MY.NET.1.65:53
TCP TTL:32 TOS:0x0 ID:39426
**SF**** Seq: 0x4C9CDBF1   Ack: 0x41B2195E   Win: 0x404
00 00 00 00 00 00                          ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

NMAP OS fingerprinting

NMAP alerts indicate OS fingerprinting is being performed and is indicative of reconnaissance efforts. This is a technique that is used to identify the operating system through identifying the behavior of the TCP/IP stack. The unique way in which different operating systems respond to various TCP flag combinations easily allows attackers to determine the OS and the version. Nmap can be obtained from http://www.nmap.org.

```
10/08-18:15:35.119165  [**] Probable NMAP fingerprint attempt [**] 132.178.218.181:3449 -> MY.NET.204.170:1632
10/27-09:57:08.723116  [**] Probable NMAP fingerprint attempt [**] 195.132.57.32:0 -> MY.NET.219.146:2529
10/27-13:50:46.676757  [**] Probable NMAP fingerprint attempt [**] 24.9.64.57:0 -> MY.NET.207.14:4389
10/06-13:38:00.767581  [**] Probable NMAP fingerprint attempt [**] 128.194.79.228:195 -> MY.NET.206.50:80
```

NMAP TCP Pings

Some of the "NMAP TCP Ping" detects maybe legitimately caused by a global Web load-balancing device, though some of the alerts to MY.NET don't exactly match this pattern. Doug McCarthy has logged this on SANS at http://www.sans.org/y2k/021401.htm. His report identifies the intrusive method in which the load-balancing device at Intel.com conducts a test to determine which pool of Web servers is closest to the client. From the sample detects, we can distill that there seems to be a distinct signature of the load-balancing device when it performs its function. The remote Web server seems to attempt to measure the RTT by sending a TCP ack to port 53 of the client's domain name server, immediately followed by an ack from 53/TCP to 53/TCP. Below taken from Doug's sample detect.

```
Jan 12 23:27:53 hostmi snort[318]: IDS28 - PING NMAP TCP:192.102.197.234:80 -> z.y.w.98:53
Jan 12 23:27:53 hostmi snort[318]: IDS28 - PING NMAP TCP:192.102.197.234:53 -> z.y.w.98:53
```

The following were logged to MY.NET that seems to follow the pattern above, however they seem to be missing the counterpart connection from port 53 to 53 connections or port 80 to 53 connections. Though the last entry in this excerpt shows the IP address (192.102.197.234) of the host identified by Doug as Intel's load-balancing device geo197a.cps.intel.com, it also does not exactly match the pattern described above.

```
10/08-00:58:17.886257  [**] NMAP TCP ping! [**] 202.187.24.3:80 -> MY.NET.1.3:53
10/08-04:41:43.174302  [**] NMAP TCP ping! [**] 209.218.228.201:80 -> MY.NET.1.8:53
10/15-10:56:50.376179  [**] NMAP TCP ping! [**] 63.104.49.126:53 -> MY.NET.1.8:53
10/16-00:25:09.081000  [**] NMAP TCP ping! [**] 63.104.49.126:53 -> MY.NET.1.8:53
10/27-10:40:20.889310  [**] NMAP TCP ping! [**] 192.102.197.234:80 -> MY.NET.1.8:53
```

The other Nmap TCP ping alerts seem to be suspicious forms of reconnaissance scanning. It is very suspicious to see connections where the source port equals the destination port and where the source port is a low port number of a WKS (Well-Known-Service). Below are alerts connections from SMTP and WWW ports.

Sport == Dport (25)

```
09/26-05:40:00.709907  [**] NMAP TCP ping! [**] 2.2.2.2:80 -> MY.NET.6.47:25
09/26-05:40:06.041838  [**] NMAP TCP ping! [**] 213.8.52.189:80 -> MY.NET.6.47:25
09/26-05:52:53.541646  [**] NMAP TCP ping! [**] 2.2.2.2:80 -> MY.NET.253.42:25
09/26-05:53:03.534575  [**] NMAP TCP ping! [**] 213.8.52.189:80 -> MY.NET.253.42:25
```

Sport == Dport (80)

```
10/04-06:46:22.064365  [**] NMAP TCP ping! [**] 202.187.24.3:80 -> MY.NET.100.165:80
10/06-09:30:45.041002  [**] NMAP TCP ping! [**] 202.187.24.3:80 -> MY.NET.6.7:80
10/10-15:45:15.963727  [**] NMAP TCP ping! [**] 12.43.88.5:80 -> MY.NET.100.165:80
10/25-04:29:51.770991  [**] NMAP TCP ping! [**] 204.155.48.3:80 -> MY.NET.253.125:80
10/27-05:26:41.367293  [**] NMAP TCP ping! [**] 203.75.25.62:80 -> MY.NET.100.165:80
```

Queso
There were three attackers that performed Queso fingerprinting and one of the three attackers 129.242.219.27 went on to attempt a WinGate attack.

- Explanation of Queso, http://www.sans.org/y2k/072500.htm
- Commonly probed port 6346, http://www.sans.org/y2k/ports.htm
- ECN and it's impact on Intrusion Detection

SMB
SMB Wildcard Alerts indicate a query for netbios information and typically can be regarded as a reconnaissance effort when coming from an external IP address. The alerts that are generated from internal hosts are false positives that are most likely triggered by a misconfigured Samba server on a Linux system.

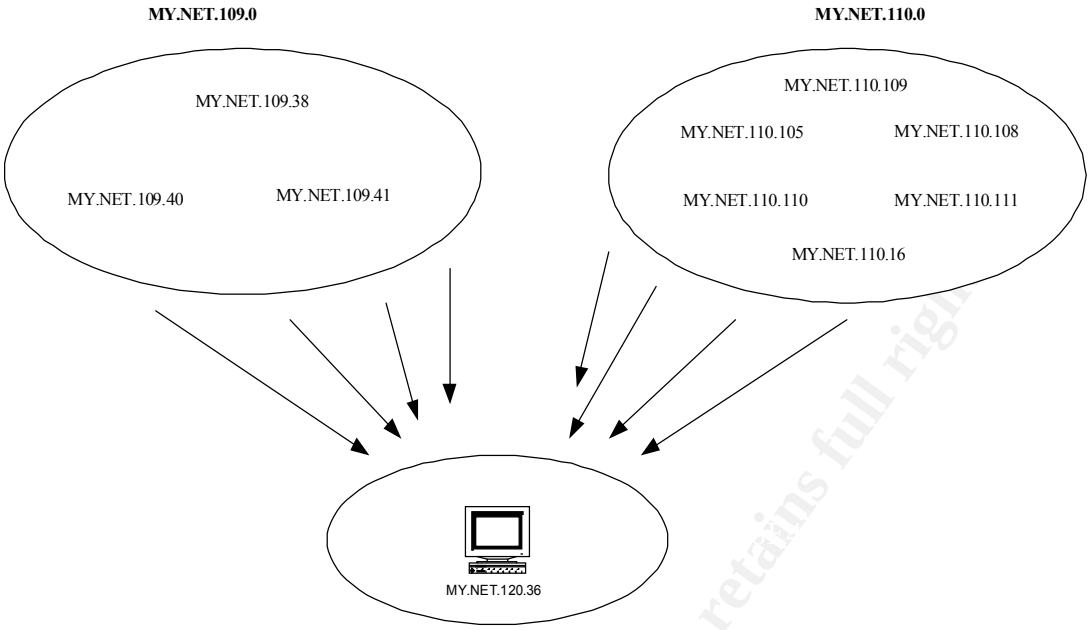
```
10/08-01:47:43.637247  [**] SMB Name Wildcard [**] 129.37.159.177:137 -> MY.NET.100.130:137
11/03-14:48:16.690486  [**] SMB Name Wildcard [**] 130.39.216.104:137 -> MY.NET.20.40:137
11/03-14:48:47.536230  [**] SMB Name Wildcard [**] 130.127.196.96:137 -> MY.NET.232.253:137
11/03-14:49:42.673280  [**] SMB Name Wildcard [**] 24.92.207.46:137 -> MY.NET.106.25:137
11/03-14:49:44.394016  [**] SMB Name Wildcard [**] 130.86.31.21:137 -> MY.NET.217.218:137
11/03-14:57:57.051490  [**] SMB Name Wildcard [**] 38.38.25.126:137 -> MY.NET.253.125:137
11/03-14:58:48.929240  [**] SMB Name Wildcard [**] 207.172.148.202:137 -> MY.NET.253.114:137
11/03-14:58:58.151156  [**] SMB Name Wildcard [**] 213.46.113.179:1035 -> MY.NET.71.38:137
11/03-15:03:22.016400  [**] SMB Name Wildcard [**] 168.143.29.9:137 -> MY.NET.60.17:137
11/03-15:03:28.000650  [**] SMB Name Wildcard [**] 24.29.206.229:137 -> MY.NET.253.134:137
11/03-15:03:29.434476  [**] SMB Name Wildcard [**] 24.29.206.229:137 -> MY.NET.253.134:137
11/03-15:04:33.980124  [**] SMB Name Wildcard [**] 213.48.182.156:1096 -> MY.NET.71.38:137
11/03-15:06:43.762884  [**] SMB Name Wildcard [**] 130.39.251.3:137 -> MY.NET.13.25:137
```

Network Mapping?? Or UDP Scanning
1586 UDP packets on source port 123 destined for host MY.NET.120.36 from 14 different source addresses on two subnets were launched in a synchronized method starting at 12:29 on October 22, 2000. This looks like a distributed probe that tries to perform network mapping to understand how the routing structure is established between different subnets. However, it seems strange that this would occur internally, since it would seem easy enough to just run traceroute to map out the routing structure.

Snort Scan Reports (multiple entries for each source host was reduced for easy viewing)

```
Oct 22 12:29:13 MY.NET.110.111:123 -> MY.NET.120.36:1632 UDP
Oct 22 12:29:16 MY.NET.110.105:123 -> MY.NET.120.36:2891 UDP
Oct 22 12:29:16 MY.NET.110.16:123 -> MY.NET.120.36:2767 UDP
Oct 22 12:29:17 MY.NET.110.109:123 -> MY.NET.120.36:1664 UDP
Oct 22 12:29:18 MY.NET.110.111:123 -> MY.NET.120.36:1082 UDP
Oct 22 12:29:18 MY.NET.110.108:123 -> MY.NET.120.36:4606 UDP
Oct 22 12:29:18 MY.NET.110.105:123 -> MY.NET.120.36:3417 UDP
Oct 22 12:29:18 MY.NET.110.110:123 -> MY.NET.120.36:3412 UDP
Oct 22 12:29:17 MY.NET.110.16:123 -> MY.NET.120.36:2918 UDP

Oct 22 12:29:13 MY.NET.109.38:123 -> MY.NET.120.36:2022 UDP
Oct 22 12:29:14 MY.NET.109.41:123 -> MY.NET.120.36:1106 UDP
Oct 22 12:29:16 MY.NET.109.40:123 -> MY.NET.120.36:1602 UDP
Oct 22 12:29:18 MY.NET.109.38:123 -> MY.NET.120.36:3605 UDP
Oct 22 12:29:18 MY.NET.109.41:123 -> MY.NET.120.36:2028 UDP
```



Attacks and Exploits

There is evidence showing active targeting using Trinoo was silently performed against several of the hosts on MY.NET causing these systems to be compromised. Post analysis of the "Snort Scan Reports" shows the different phases of the Trinoo attack being launched. The Snort sensor did not detect the attack, Trinoo because most likely it was not configured with IDS signature for this popular attack.

There is also evidence of intruders targeting subnet 70 for problems associated with ICMP broadcasts. Potentially, this can lead to denial of service and can be launched in a large scale through a distributed attack from multiple spoofed locations. Any previous outages to services on subnet 70 can be historically correlated against the dates of the logged broadcast attacks.

Additionally, Wingate attempts, ftp exploits, and Back Orifice have been detected.

Trinoo Attacks

The activity of a DDOS tool known as Trinoo was identified upon manual inspection of the log contents of the "Snort Scan Reports". The Snort sensor did not catch this and detect this as an alarm. Ten hosts within MY.NET seem to have been compromised since at one point they had port 1524/TCP running. From the Snort Scan Reports we see that the attacker was able to successfully login to MY.NET.253.114 to begin sending attack commands and has been able to make the victim host respond to the intruder's commands.

The UNIX version of Trinoo relies on several TCP and UDP based ports – 1523/TCP, 27665/TCP, 274444/UDP, and 31335/UDP, so investigation requires filtering out log entries that fit these criteria. In a nutshell, Trinoo attacks break down into 2 phases – preparation work and launching the attack. Below is an outline of the steps taken in each phase of the attack.

Preparation phase

1. Intruder first finds hosts vulnerable to RPC services 'statd', 'cmsd' and 'ttdbserverd' CERT IN-99-04
2. Compromises these systems
3. Sets up listening shell port on 1524/TCP
4. Contacts host on 1524/TCP to install Trinoo daemon and rootkit

Attack phase of Trinoo

1. Telnets to port 27665/TCP, enters password "betaalmostdone"
2. Send command via 27444/UDP, and receives responses to with 31335/UDP

Preparation Phase, Shell port being contacted

```
Oct  8 07:14:53 202.100.34.235:3926 -> MY.NET.70.121:1524 SYN **S*****
Oct  8 07:14:56 202.100.34.235:3926 -> MY.NET.70.121:1524 SYN **S*****
Oct 14 12:29:11 195.34.28.117:3162 -> MY.NET.97.59:1524 SYN **S*****
Oct 14 12:29:24 195.34.28.117:3236 -> MY.NET.97.59:1524 SYN **S*****
Oct 16 08:56:42 213.186.141.243:2122 -> MY.NET.253.114:1524 SYN **S*****
Oct 16 11:01:46 212.34.32.91:1150 -> MY.NET.221.82:1524 SYN **S*****
Oct 19 19:45:26 209.148.79.140:4814 -> MY.NET.98.168:1524 SYN **S*****
Oct 21 08:25:18 209.23.1.66:3292 -> MY.NET.204.26:1524 SYN **S*****
Oct 21 08:25:22 209.23.1.66:3292 -> MY.NET.204.26:1524 SYN **S*****
Oct 23 22:44:04 141.157.98.201:62419 -> MY.NET.1.6:1524 SYN **S*****
Nov  4 10:40:10 209.195.145.34:1730 -> MY.NET.201.194:1524 SYN **S*****
Nov  7 20:27:27 24.6.151.155:63054 -> MY.NET.162.36:1524 SYN **S*****
```

Beginning of Attack Phase

```
Oct  8 07:15:32 202.100.34.235:4296 -> MY.NET.70.121:27665 SYN **S*****
Oct  8 07:15:33 202.100.34.235:4296 -> MY.NET.70.121:27665 SYN **S*****
Oct  8 07:15:36 202.100.34.235:4296 -> MY.NET.70.121:27665 SYN **S*****
```

Sending commands!!

```
Sep 28 13:26:36 24.18.90.197:38180 -> MY.NET.253.114:27444 UDP
```

Responding to commands!!!

```
Sep 28 13:26:37 24.18.90.197:38180 -> MY.NET.253.114:31335 UDP
```

The details of Trinoo are described in a document titled, "Distributed Denial of Service Attack Tools: trinoo and wintrinoo".

ICMP Broadcast Problems and SMURF Attacks

Subnet 70 is vulnerable to DOS caused by ICMP amplification. An IP directed broadcast is a datagram which is sent to the broadcast address of a subnet to which the sending machine is not directly attached. The broadcast address varies depending on the netmask of the subnet. For a C Class subnet (24bit mask) the broadcast address is 0 and 255. The directed broadcast is routed through the network as a unicast packet until it arrives at the target subnet, where it is converted into a link-layer broadcast. Because of the nature of the IP addressing architecture, only the last router in the chain, the one that is connected directly to the target subnet, can conclusively identify a directed broadcast.

IP directed broadcasts are used in the extremely common and popular "smurf" denial of service attack, and can also be used in related attacks. In a "smurf" attack, the attacker sends ICMP echo requests from a falsified source address to a directed broadcast address, causing all the hosts on the target subnet to send replies to the falsified source.

The following are portions of the alert log showing evidence of active targeting. The intent of this technique is to cause denial of service (DOS). By sending a continuous stream of such requests, the attacker can create a much larger stream of replies, which can completely inundate the host whose address is being falsified. Additionally, as a consequence it can also choke up the network bandwidth due to the overwhelming response. This attack method can also be used in distributed denial of service (DDOS).

Evidence of targeting vulnerabilities with ICMP broadcasts to networks

```
11/08-16:29:07.812696  [**] Broadcast Ping to subnet 70 [**] 213.154.131.131 -> MY.NET.70.255

11/08-18:17:33.724198  [**] Broadcast Ping to subnet 70 [**] 193.226.60.179 -> MY.NET.70.255

10/22-17:28:52.521917  [**] Broadcast Ping to subnet 70 [**] 63.27.120.204 -> MY.NET.70.255

10/23-10:59:37.143954  [**] Broadcast Ping to subnet 70 [**] 129.186.67.59 -> MY.NET.70.255
```

Wingate

This is a Windows based proxy that is often incorrectly configured and it has become a common method for hackers to tunnel and forward their attacks through the guise of the proxy address. If any of the host on MY.NET in these alerts are proxy servers, it is recommended that an analysis of these files be performed to further pursue event correlation. Additionally, it might be a good idea to check to see if any of the hosts are identified on http://dfdfdffsd which has a listing of proxy servers.

The CVEs listed at http://cve.mitre.org explaining the vulnerabilities: CVE-1999-0290, CVE-1999-0291, CVE-1999-0441, and CVE-1999-0494.

WU_FTP exploit

FTP is one of the most vulnerable Internet services because of the permissions provided to external clients to write and read to and from the system's hard drive. To compound this problem, FTP like many other applications suffers the problems of unprotected memory spaces that allow the corrupt execution of writing past the array's boundary declared in a routine. There were very few occurrences of these types of attack, but this should be monitored since the severity of this attack is very lethal.

- Smashing the Stack for Fun and Profit, issue #49
- CA-99-03-FTP-Buffer-Overflows
- CA-99-13 Multiple Vulnerabilities in WU-FTPD

Back Orifice

From the alert logs we see that intruders are trying to contact hosts on MY.NET for Back Orifice, which is known to run on port 31337. However, we cannot say with certainty, that these hosts have been compromised through this Trojan attack, since we do not see any signs of evidence showing that the hosts are responding back in the typical "client/server" session pattern. To verify signs of compromise it requires not only signatures detecting connections in-bound to MY.NET but also Back Orifice signatures that would detect our hosts communicating back to the attacker. Some of the following rules below should be included to make correlations with the alerts detected for in-bound connections to port 31337.

- Information can be found from Internet Security's Systems documenting the Back Orifice 2000 Security Advisory
- Listing of Back Orifice as common security vulnerabilities, CAN-1999-0660

Napster stuff

The alerts found for "Watchlist 000220 IL-ISDNNET-990517" at first glance seem to indicate some Napster like file sharing application in progress. However upon closer examination of these alerts, they don't seem to exhibit the typical behavior after connecting to a Napster server on port 6699, there is a usually a flurry of inbound connections to the port 6699 to the client that initially made a connection. Below is a sample of typical Napster traffic provided by Crist Clark.

Care should be taken to better monitor Napster like applications due to the many vulnerabilities described in CAN-2000-0281, and CAN-2000-0412. In the document NAPSTER - Should You Be Worried About It?, it reports that Napster can be set to user defined ports and also informs us that add-on utilities like can Wrapster , any type of file sharing can be performed. Napster by default only allows mp3 file types to be shared.

Typical Napster Traffic

```
2Nov2000  9:31:35 accept >qfe3  tcp 192.168.XXX.186:1412 -> 172.144.30.193:6699 44 (XXX.XXX.248.142:54446 ->
172.144.30.193:6699)
2Nov2000  9:31:44 accept >qfe3  tcp XXX.XXX.152.239:1812 -> 203.96.106.137:6699 44 (XXX.XXX.248.142:54545 ->
203.96.106.137:6699)
2Nov2000  9:35:44   drop >hme0  tcp 200.28.48.106:4020 -> XXX.XXX.248.142:6699 44
2Nov2000  9:41:25   drop >hme0  tcp 62.36.149.102:1104 -> XXX.XXX.248.142:6699 44
2Nov2000  9:44:12   drop >hme0  tcp 198.213.203.57:1133 -> XXX.XXX.248.142:6699 48
2Nov2000  9:44:27   drop >hme0  tcp 212.120.103.108:1273 -> XXX.XXX.248.142:6699 48
2Nov2000  9:47:51   drop >hme0  tcp 195.223.93.163:1185 -> XXX.XXX.248.142:6699 48
2Nov2000  9:52:21 accept >qfe3  tcp XXX.XXX.153.168:1292 -> 62.227.192.50:6699 64 (XXX.XXX.248.142:13156 ->
62.227.192.50:6699)
2Nov2000  9:56:31 accept >qfe3  tcp XXX.XXX.153.196:1148 -> 4.4.58.52:6699 44 (XXX.XXX.248.142:14296 ->
4.4.58.52:6699)
2Nov2000 10:02:32   drop >hme0  tcp 63.16.57.29:1180 -> XXX.XXX.248.142:6699 48
2Nov2000 10:10:08   drop >hme0  tcp 212.14.119.159:1458 -> XXX.XXX.248.142:6699 48
```

Alerts Originating from Inside

The following represent a reconnaissance technique and the other is an attack against the UNIX based printing daemon. Both of these activities were inside attacks.

Internal SNMP Probes
Detects show that some internal hosts were actively probing the device MY.NET.101.192. 468 alerts were generated by 10 internal addresses. Could it be possible that this device is a critical router or switch running SNMP and some of the attackers are trying to cause some denial of service by changing the status of the interfaces from up to down?

- Using SNMP for Reconnaissance.

LPRng Exploits
The host MY.NET101.142 should be investigated to see if there are any user or users that can be linked to the attack to MY.NET.100.3. The vulnerabilities of the UNIX printing daemon are described in the links listed below.

```
11/19-14:06:40.485631  [**] connect to 515 from inside [**] MY.NET.101.142:1020 -> MY.NET 100.3:515
11/19-14:06:40.485989  [**] connect to 515 from inside [**]MY.NET.101.142:1020 -> MY.NET 100.3:515
11/19-14:06:40.519519  [**] connect to 515 from inside [**]MY.NET.101.142:1020 -> MY.NET 100.3:515
11/19-17:06:39.279618  [**] connect to 515 from inside [**]MY.NET.101.142:1022 -> MY.NET 100.3:515
```

- Alert: Increased probes to TCP port 515
- LPRng-redhat7-overflow-rdC--IDS456
- LPRng-redhat7-overflow-security.is -- IDS457

False Positive Alerts

The following three alerts seem to have triggered false positive alerts that are typically associated with RPC high port access, SMB, and network mapping using UDP. The reasons for determining these detects as false alarms are described in each section.

ICQ Activity Reported as RPC High Port Access

These detects seem to be false positives and show low possibility of being an attack. According to the logs, the attack against port 37721 ruserd continues on for days. Exploits to SUN RPC vulnerabilities typically try to overflow buffers does not take days to execute. The possibility of that this is an attack is further decreased because addresses in 205.188.153.0 have been known to host ICQ, a popular instant messaging software.

```
10/03-22:49:15.221332  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771
10/03-22:50:30.265615  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771
10/03-22:52:49.396229  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771
10/03-22:56:29.662646  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771

10/04-01:39:50.781589  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771
10/04-01:52:23.195818  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771
10/04-02:07:23.823281  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771
10/04-02:13:23.225897  [**] Attempted Sun RPC high port access [**] 205.188.153.116:4000 -> MY.NET.225.210:32771
```

SMB False Alerts

As mentioned earlier, it seems that one internal host seems to be misconfigured triggering some false positives to SMB Alerts.

```
10/10-17:19:18.303837  [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
10/28-14:09:06.007257  [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137 11/14-
17:03:56.281011  [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
11/19-09:28:58.588194  [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
```

Side-Effects of Global Load-Balancing Devices

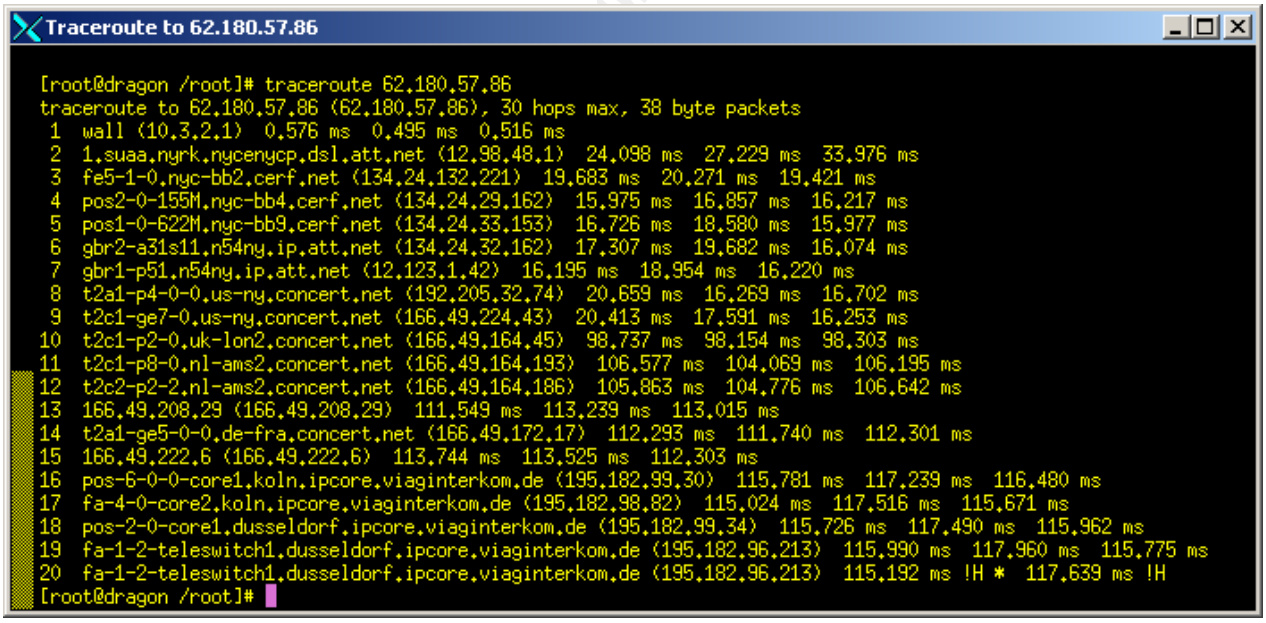
Since both source addresses are ISP entities in Europe, it may possibly be that a global load-balancing device made these measurements.

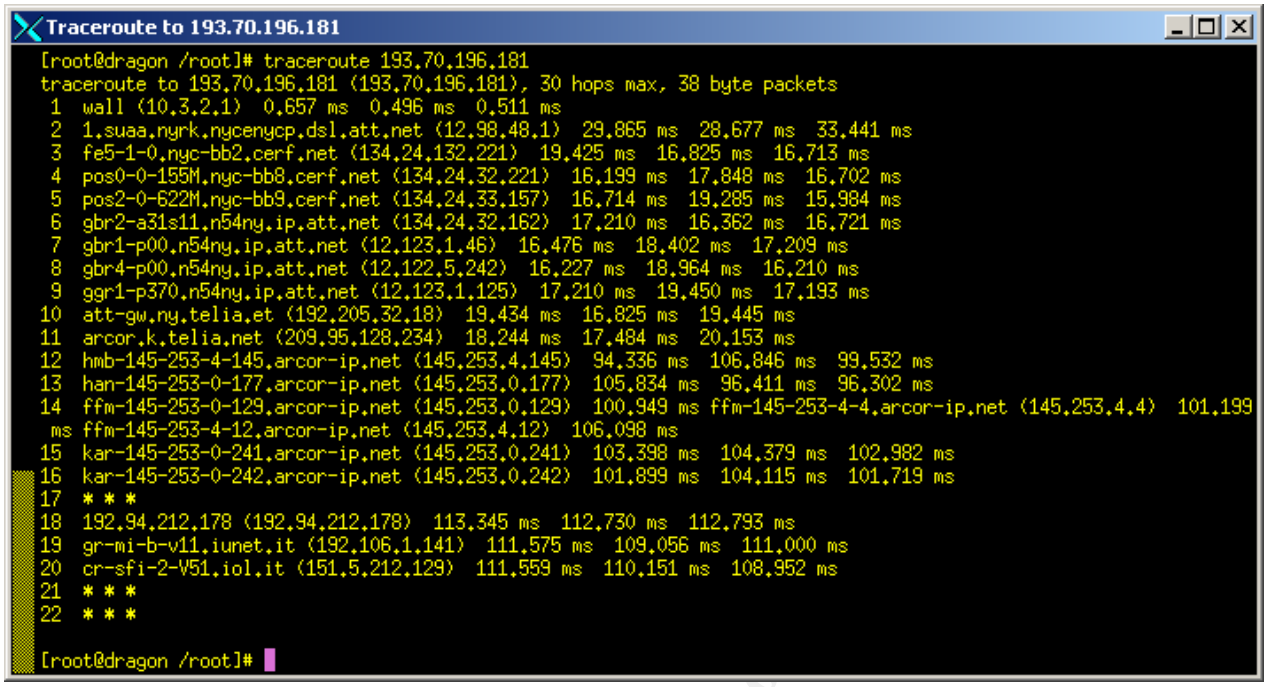
The following logs also show the similar type signature of reconnaissance efforts of the network mapping technique described earlier. Once again a UDP based scan is performed simultaneously from two IP addresses from two different domains to the same destination at relatively the same time (28 second time difference). Both the source and destination UDP ports are using ephemeral port ranges and they seem to be incrementing, perhaps this is a legitimate attempt to perform distance vector checking.

```
Sep 28 05:04:39 62.180.57.86:27017 -> MY.NET.208.118:2567 UDP
Sep 28 05:04:39 62.180.57.86:27016 -> MY.NET.208.118:2568 UDP
Sep 28 05:04:39 62.180.57.86:27011 -> MY.NET.208.118:2573 UDP
Sep 28 05:04:39 62.180.57.86:27020 -> MY.NET.208.118:2566 UDP
Sep 28 05:04:39 62.180.57.86:27013 -> MY.NET.208.118:2571 UDP
Sep 28 05:04:39 62.180.57.86:27012 -> MY.NET.208.118:2572 UDP
Sep 28 05:04:39 62.180.57.86:27014 -> MY.NET.208.118:2570 UDP

Sep 28 05:05:07 193.70.196.181:28005 -> MY.NET.208.118:1902 UDP
Sep 28 05:05:07 193.70.196.181:28003 -> MY.NET.208.118:1904 UDP
Sep 28 05:05:07 193.70.196.181:28002 -> MY.NET.208.118:1905 UDP
Sep 28 05:05:07 193.70.196.181:28001 -> MY.NET.208.118:1906 UDP
Sep 28 05:05:07 193.70.196.181:27019 -> MY.NET.208.118:1907 UDP
Sep 28 05:05:07 193.70.196.181:27017 -> MY.NET.208.118:1909 UDP
Sep 28 05:05:07 193.70.196.181:27018 -> MY.NET.208.118:1908 UDP
Sep 28 05:05:07 193.70.196.181:27015 -> MY.NET.208.118:1911 UDP
Sep 28 05:05:07 193.70.196.181:28004 -> MY.NET.208.118:1903 UDP
```

Traceroutes to 62.180.57.86 traces back to a reserved block of addresses in Germany owned by viaginterkom.de. The 192.70.196.0 – 192.70.196.255 on Arin records shows it is owned by some entity called USC-ISI in Del Ray, California; however, traceroute follows this address back to Italia Online services in Italy. Further detail regarding the domains follows in the Appendix section of this report.





List of Attacks by Country

The chart below shows that the 3 countries that contributed the highest level of alerts were Israel, USA, and Italy. Other countries originating the alerts are as follows:

Country	Total number of probes
Israel	29,263
USA	23,544
Italy	10,491
Korea	8,222
China	7,509
Netherlands	5,665
Australia	3,545
France	3,399
Finland	3,295
Spain	2,338
Hong Kong	1,584
Argentina	1,105

The majority of these attacks originated from these domains, which are all ISPs except for unipr.it and ac.cn, which are universities in Italy and China. Detailed information regarding the domains can be found in the Appendix section of this report.

List of Domains Originating Attacks

Domain name	Network Blocks	Country
bezeqint.net	212.179.0.0	Israel
Unipr.it	160.68.0.0	Italy
Bellsouth.net	208.61.4.0	USA
Ac.cn	159.226.0.0	China
Pacbell.net	63.195.56.0-63.195.59.0	USA

List of Top Talkers

This list was compiled based on the source addresses that generated more than 300 or more alerts.

Source Address	Number of Alerts	Percentage	Country
160.78.49.191	7,199	6.51	Italy
208.61.4.207	6,635	6.00	USA
159.226.45.3	6,297	5.70	China
212.179.95.5	6,117	5.53	Israel
209.92.40.32	4,967	4.49	USA
212.179.27.6	4,011	3.63	Israel
212.179.79.2	3,950	3.57	Israel
212.179.44.115	3,938	3.56	Israel
63.195.56.20	3,897	3.53	USA
130.89.229.48	3,860	3.49	Netherlands
210.113.89.200	3,572	3.23	Korea
203.32.161.197	3,545	3.21	Australia
213.41.69.52	3,399	3.08	France
193.64.114.10	3,295	2.98	Finland
195.103.69.159	3,292	2.98	Italy
210.101.101.11	2,582	2.34	Korea
212.0.107.107	2,338	2.12	Spain
211.46.110.81	2,068	1.87	Korea
63.193.210.208	1,883	1.70	USA
212.179.72.226	1,591	1.44	Israel
143.89.13.3	1,584	1.43	Hong Kong
128.2.81.133	1,569	1.42	USA
63.167.58.13	1,531	1.39	USA
212.179.41.24	1,353	1.22	Israel
159.226.91.20	1,212	1.10	China
24.7.227.215	1,148	1.04	USA
163.10.19.34	1,105	1.00	Argentina
212.187.21.156	1,085	0.98	Amsterdam
212.179.45.81	950	0.86	Israel
212.179.66.2	729	0.66	Israel
212.179.44.66	667	0.60	Israel
212.179.29.170	648	0.59	Israel
205.188.153.10	628	0.57	USA
212.179.95.26	625	0.57	Israel
212.179.7.58	589	0.53	Israel
212.179.30.113	579	0.52	Israel
212.179.15.122	564	0.51	Israel
205.188.153.10	517	0.47	USA
212.179.50.77	505	0.46	Israel
212.179.24.136	475	0.43	Israel
212.179.56.5	439	0.40	Israel
205.188.153.11	435	0.39	USA
212.179.23.95	416	0.38	Israel
212.179.45.241	402	0.36	Israel
212.179.58.191	366	0.33	Israel
212.179.95.45	349	0.32	Israel
205.188.153.10	334	0.30	USA

Defensive Recommendations

Based on the vulnerabilities discovered at MY.NET through analysis of the Snort alert and log files from MY.NET, here is a summary of some of the major security recommendations that should be implemented immediately.

- Update Snort IDS configuration file, it should contain all of the most common attack signatures
- Check Snort rule set signatures to provide more granular examination of SYN-FIN alerts
- Upgrade or install firewall if not installed and perform aggressive port scanning tests against firewall policy
- Deny all access from hosts in bezeqi.net and ac.cn domains to MY.NET if not required for business
- Turn off ICMP broadcast and or block all inbound ICMP packets to MY.NET
- Installation of virus detection utilities on the workstations company-wide if it has not been done

1. Tweaking Snort

The Snort IDS signature file should be upgraded to the latest version, which can be obtained from Snort's Home Page. The IDS signature should include detection for most of the common backdoors such as Trinoo, which went undetected previously at MY.NET. Be sure to include signature files that would detect out-bound Back Orifice traffic. Additionally, the Snort rules need to provide more granular examination of SYN-FIN scans, which may show some more details in the stealth activity or in new types of SYN-FIN scans.

Due to the large volume of output generated by an IDS tool like Snort, it is recommended that the alerts and log files be outputted to a binary file format (To generate logs and alerts in a binary format run Snort with the–b option). Doing this will conveniently save the binary file with the date and time in the following format **snort-0220@1239**. This also makes it very easy to access the contents either using Snort or tcpdump with the –r, read option set. It is also possible to compile Snort so that it can allow the sensor to log to a centralized SQL database such as MYSQL or Postgress. Both methods make it easy to review the data and perform different queries against the data with relative ease.

Snort IDS Signature for Trinoo

```
alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"IDS187 - DDoS - Trin00:DaemontoMaster(PONGdetected)"; content:"PONG";)
alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"IDS186 - DDoS - Trin00:DaemontoMaster(messagedetected)"; content:"l44";)
alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"IDS185 - DDoS - Trin00:DaemontoMaster(*HELLO*detected)";
content:"*HELLO*";)
alert tcp $EXTERNAL_NET any -> $HOME_NET 27665 (msg:"IDS196 - DDoS - Trin00:Attacker to Master default startup pass
detected!";flags:PA; content:"betaalmostdone";)
alert tcp $EXTERNAL_NET any -> $HOME_NET 27665 (msg:"DDoS - Trin00 Attacker to Master defaultr.i.passdetected!";flags:PA;
content:"gOrave";)
alert tcp $EXTERNAL_NET any -> $HOME_NET 27665 (msg:"DDoS - Trin00 Attacker to Master-default mdie pass detected!";flags:PA;
content:"killme";)
alert udp $EXTERNAL_NET any -> $HOME_NET 27444 (msg:"IDS197 - DDoS - Trin00:MastertoDaemon(defaultpassdetected!)";
content:"l44adsl";)
```

Additional Snort Rules to Detect Compromised Hosts Trying to Connect Back to the Attacker using BackOrifice

```
alert tcp $HOME_NET 31337 -> $EXTERNAL_NET any (msg:"IDS189 - BACKDOOR ACTIVITY-Possible Backorifice";flags:SA;)
alert tcp $HOME_NET 54321 -> $EXTERNAL_NET any (msg:"IDS189 - BACKDOOR ACTIVITY-Possible BackOrifice 2000"; flags:SA;)
alert tcp $HOME_NET 54320 -> $EXTERNAL_NET any (msg:"IDS189 - BACKDOOR ACTIVITY-Possible BackOrifice 2000"; flags:SA;)
alert udp $HOME_NET 54321 -> any any (msg:"IDS189 - BACKDOOR ACTIVITY-Possible Back Orifice 2k";)
```

2. Network Protection

Most of the probes into MY.NET could have been prevented with any firewall that can perform stateful inspection correctly and by locking down inbound access into the environment. To begin with the entire domain BEZEQINT.NET and AC.CN should be denied ALL access into MY.NET if this does not impact MY.NET from performing business. Other external hosts should also be prohibited from sending ICMP and UDP protocols in-bound, since most intruders take great advantage of these protocols and since many administrators have underestimated the power of these "relatively harmless" protocols. If ping and or traceroute are absolutely required, the security administrator should consider, isolating this access to a restricted network. Again if ICMP are absolutely

necessary, it is also imperative to prevent ICMP broadcasts by configuring routers and hosts on that particular segment to ignore the broadcasts. On the routing device of subnet 70 turn off ICMP directed broadcasts. If there are Solaris hosts their kernel can be tuned to ignore all broadcasts as follows.

ndd -set /dev/ip ip_forward_directed_broadcasts 0
ndd -set /dev/ip ip_respond_to_echo_broadcast 0

3. Buffer Overflow Protection

There are always new problems with software that are linked to buffer overflow problems, therefore, it is always a good idea to keep up with the latest security alerts which can be obtained from http://www.sans.org or from http://www.cert.org. Also always check to see if the vendor has provided a patch or upgrade.

Sun Microsystems operating system, Solaris allows protection against stack based buffer overflows, by modifying the kernel parameters. Add the following to /etc/system:

set noexec_user_stack=1
set noexec_user_stack_log =1

The first line prevents overflows, while the second line logs user attempts to overflow the buffer.

Summary analysis

We have evidence showing that the firewall or screening router in place protecting the external borders of MY.NET is ineffectual at blocking inbound connections and leaks too information to intruders. The alerts and logs show a variety of TCP and UDP stealth scans that have allowed intruders to perform reconnaissance.

There is also evidence that shows some hosts have been compromised by the backdoor attack Trinoo and that subnet 70 of MY.NET is susceptible to broadcast storms, which can result in denial of service. There has been some other internal based attack that requires some follow up to see if either someone within MY.NET is causing these attacks or if it is possible that these internal hosts have been compromised by intruders and are furthering their attack. Lastly, there are some internal users that seem to be using some vulnerable file sharing utilities like Napster that should be evaluated to determine if they pose a potential risk to the business.

IDS strategies require on-going maintenance to ensure latest attacks will be detected and that the logs are carefully archived for future analysis and reference. As we have seen, we cannot be complacent by solely relying on the automated triggering of alerts or the lack of triggered alerts.

Assignment 4 - Analysis Process

The analysis was performed on a SMP Linux 2.3 box with 2 x 500 MHZ Celerons and 128MGs of RAM. Using this system resource it took approximately 2hrs to run snortsnarf.pl, a PERL script against the data set in a single pass. Because of tremendous size of the data, and because some of the formats provided for analysis were not acceptable to snortsnarf.pl, it was necessary to write some additional PERL scripts to perform further analysis.

Utilized SnortSnarf v0116101.1 to analyze data which is available at www.silicondefense.com/snortsnarf/main.html.

Log Processing Procedures using snortsnarf.pl

1. Merged all files together into one file.
2. Converted obfuscated naming for the scanned network from MY.NET to 255.255 so that snortsnarf would be able to track the number of attacks from source and destinations, otherwise snorfsnarf will not output this data.
3. Ran snortsnarf as follows: ./snortsnarf –dns singlefile. The –dns was used so that a DNS lookup.

Log Processing Procedures using home spun PERL scripts

1. Merged all "Snort Alert Report" files into one file and searched for IDS alert.
2. Categorized similar alerts into separate files according to IDS alerts.
3. Depending on garden variety of alert, applied the appropriate analytical process listed below.

Analytical Process

1. View variety of attack signatures identified
2. View source of attackers
3. View victim of attacks
4. Reverse engineer attacks. Group alerts based on reconnaissance probes and actual attacks. Volume doesn't mean everything. Something more malicious may only leave a small footprint.
   Filter down → Attackers = (src of attacks)
   Probers =Reconn srcs (SYN-FINs : TCP pings : Queso : TCP Stack fingering)
   Targeted attackers = (Attackers && Probers)
5. Look for distributed or coordinated attacks based on seed of timestamp and directed attack against a specific address or network.
6. Look for sequence patterns in attacks that uses multiple ports
7. Determine level of intrusion into network
8. Identity network weakness and hosts aiding in problems or attacks
9. Identify compromised hosts

Appendix -- Information on Domain Registration of Domains Originating Attacks

---
AC.CN

[whois.arin.net]

The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
P.O. Box 2704-10,
Institute of Computing Technology Chinese Academy of Sciences
Beijing 100080, China

Netname: NCFC
Netblock: 159.226.0.0 - 159.226.255.255

Coordinator:
Qian, Haulin (QH3-ARIN) hlqian@NS.CNC.AC.CN
+86 1 2569960

Domain System inverse mapping provided by:

NS.CNC.AC.CN	159.226.1.1
GINGKO.ICT.AC.CN	159.226.40.1

Record last updated on 25-Jul-1994.
Database last updated on 15-Feb-2001 19:15:24 EDT.

BEZEQINT.NET

Initial server used for this query: whois.networksolutions.com

Registrant:
Bezeq International (BEZEQINT2-DOM)
40 Hashacham St.
Petach Tikva, Israel 49170
IL
IL

Domain Name: BEZEQINT.NET

Administrative Contact:
Pinko, Nati (NP2484) hostmaster@ISDN.NET.IL
bezeq-int-isdnnet
40 Hashacham st
Petach Tikva
49170
IL
3-9279961 (FAX) 3-9279961
Technical Contact:
Peer, Tomer (TP5909) hostmaster@BEZEQINT.NET
ISDN Net-Bezeqint
Hashacham 40
Petah-tikva
IL
49170
IL

972-3-9257778 (FAX) 972-3-9220135
Billing Contact:
Bezeq International Billing Dep. (BI3752-ORG) elil@BEZEQINT.CO.IL
Bezeq International
40 hashacham Street
Petach-Tikva
ISRAEL
972-3-9257303Fax- 972-3-9257369 Fax- - 972-3-9257369

Record last updated on 05-Nov-2000.
Record expires on 04-Nov-2010.
Record created on 04-Nov-1998.
Database last updated on 28-Jan-2001 17:17:39 EST.

Domain servers in listed order:

NS1.BEZEQINT.NET 192.115.106.10
NS2.BEZEQINT.NET 192.115.106.11

BellSouth.net Inc. (NETBLK-BELLSNET-BLK7)

---

301 Perimeter Center North, Suite 400
Atlanta, GA 30346
US

Netname: BELLSNET-BLK7
Netblock: 208.60.0.0 - 208.63.255.255
Maintainer: BELL

Coordinator:
Geurin, Joe (JG726-ARIN) ipadmin@bellsouth.net
678-441-7800 (FAX) 678-441-6968

Domain System inverse mapping provided by:

NS.BELLSOUTH.NET 205.152.0.5
NS.ATL.BELLSOUTH.NET 205.152.0.20

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

=====

NOTE: For abuse issues, please email abuse@bellsouth.net.

=====

Record last updated on 12-Sep-2000.
Database last updated on 26-Jan-2001 18:41:43 EDT.

Unipr.it

[whois.arin.net]

Centro di Calcolo di Ateneo (NET-PARMANET1)
Centro di Calcolo di Ateneo
Universita` di Parma
Viale Delle Scienze
43100 PARMA - ITALIA

Netname: PARMANET
Netblock: 160.78.0.0 - 160.78.255.255

Coordinator:
Fausto, Lina (LF112-ARIN) FAUSTO@IPRUNIV
+39 521 580392

Domain System inverse mapping provided by:

SERVER.FIS.UNIPR.IT	192.135.11.20
CAIO.CCE.UNIPR.IT	160.78.48.10

[whois.nic.it]

domain: unipr.it
x400-domain: c=it; admd=garr; prmd=unipr;
org: Universita' degli Studi di Parma
descr: Universita' degli Studi, didactic and scientific research
admin-c: LF112
tech-c: DM1124-RIPE
tech-c: RA492-RIPE
postmaster: LF112
postmaster: DM1124-RIPE
zone-c: RA492-RIPE
zone-c: MG1193-RIPE
nserver: 160.78.48.10 caio.cce.unipr.it
nserver: 192.135.11.20 server.fis.unipr.it
dom-net: 160.78.0.0
remarks: fully managed
mnt-by: GARR-MNT
created: before 960129
changed: 19930729
source: IT-NIC

person: Fausto Lina
address: Dipartimento di Fisica
address: Viale delle Scienze
address: I-43100 Parma
address: Italy
phone: +39 0521 580392
fax-no: +39 0521 580469
e-mail: fausto@ipruniv.cce.unipr.it
nic-hdl: LF112
changed: 930729
source: IT-NIC

person: Daniela Marmiroli
address: Centro Calcolo di Ateneo
address: Parco Area delle Scienze 17/a
address: Parma
address: Italy
phone: +39 521 905483

fax-no: +39 521 905469
e-mail: daniela@ipruniv.cce.unipr.it
nic-hdl: DM1124-RIPE
changed: hostmaster@nic.it 980921
source: IT-NIC

person: Roberto Alfieri
address: Istituto Nazionale di Fisica Nucleare
address: Sezione di Parma
address: Viale delle scienze
address: I-43100 Parma
address: Italy
phone: +39 521 905278
fax-no: +39 521 905223
e-mail: roberto.alfieri@pr.infn.it
nic-hdl: RA492-RIPE
changed: hostmaster@nic.it 980921
source: IT-NIC

person: Massimo Golinelli
address: Viale Le Scienze, 78
address: 43100 PARMA
phone: +39 521 905470
fax-no: +39 521 905369
e-mail: massimo@unipr.it
nic-hdl: MG1193-RIPE
changed: hostmaster@nic.it 980921
source: IT-NIC

```
inetnum:      62.180.0.0 - 62.180.255.255
netname:      DE-VIAG-980710
descr:        VIAG-INTERKOM
descr:        PROVIDER
country:      DE
admin-c:      VIAG1-RIPE
tech-c:       VIAG1-RIPE
status:       ALLOCATED PA
mnt-by:       RIPE-NCC-HM-MNT
mnt-lower:    VIAG-MNT
changed:      hostmaster@ripe.net 19980710
changed:      hostmaster@ripe.net 19990428
changed:      hostmaster@ripe.net 19990616
changed:      hostmaster@ripe.net 19990628
changed:      hostmaster@ripe.net 19990917
source:       RIPE

route:        62.180.0.0/16
descr:        DE-VIAG-980710
origin:       AS8472
mnt-by:       VIAG-MNT
changed:      Kurt.Kayser@viaginterkom.de 19980710
source:       RIPE

role:         VIAG Interkom-Service-Center
address:      VIAG Interkom GmbH & Co
address:      Network Service Center
address:      Mergenthalerallee 6-8
address:      D-65760 Eschborn
address:      Germany
```

```
phone:        +49 69 3307 6611
fax-no:       +49 69 3307 1111
e-mail:       hostmaster@viaginterkom.de
trouble:      SPAM/COMPLAINTS to: abuse@viaginterkom.de
trouble:      FAULTS to: noc@viaginterkom.de
trouble:      Call: 0180 5515055 for more information
trouble:      Looking for services? -> http://www.viaginterkom.de
trouble:      SPAM/COMPLAINTS mails to the persons below will be ignored.
admin-c:      KM2133-RIPE
tech-c:       SR1985-RIPE
tech-c:       DAVE2-RIPE
nic-hdl:      VIAG1-RIPE
remarks:      last change just updated trouble fields.
notify:       hm-dbm-msgs@ripe.net
notify:       ripe@planning.viaginterkom.de
mnt-by:       VIAG-MNT
changed:      hostmaster@viaginterkom.de 19990610
changed:      hostmaster@viaginterkom.de 19990616
changed:      hostmaster@viaginterkom.de 19990909
changed:      dave.pratt@viaginterkom.de 20010215
source:       RIPE
```

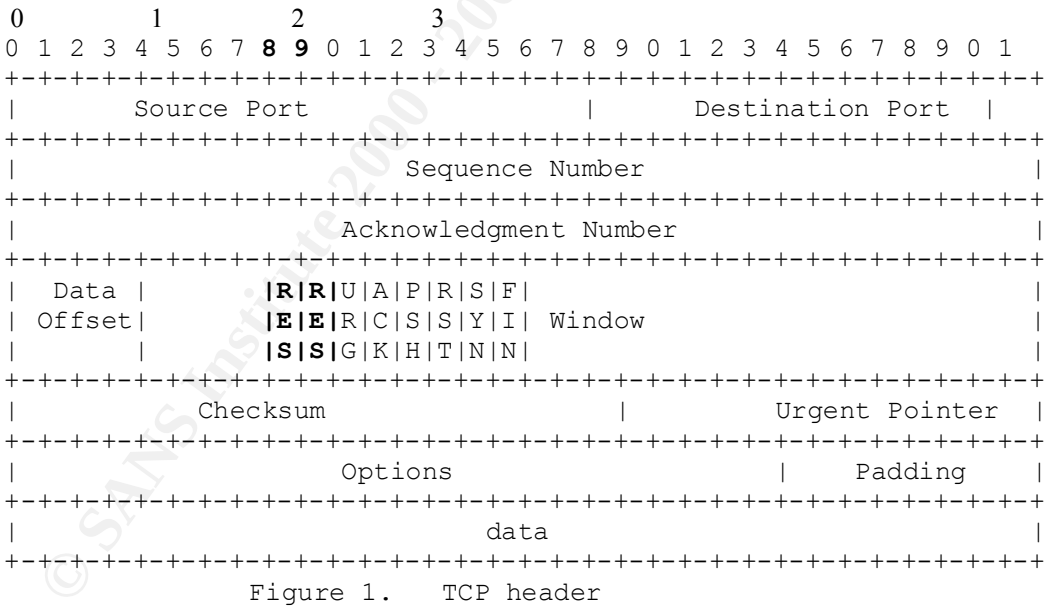



Figure 1. TCP header

.....

Last Updated: July 25th, 2017

Upcoming Training

SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced