SANS GIAC Certified Intrusion Analyst Submission of Practical Assignment using Monterey NS2000 format

by Gustavo Monserrat

11/22/00

Network detects

Detect #1

```
11:13:59.802343 24.113.18.56.3490 > my.box.3128: S 365882618:365882618(0) win 44620 (DF)(ttl
53, id 36549)
11:13:59.808225 24.113. 18.56.3491 > my.box.8080: S 365946237:365946237(0) win 44620 (DF)(ttl
53, id 36550)
11:13:59.822294 24.113.18.56.3492 > my.box.80: S 365990955:365990955(0) win 44620 (DF)(ttl
53, id 36551)
11:14:02.750984 24.113.18.56.3491 > my.box.8080: S 365946237:3659 46237(0) win 44620 (DF)(ttl
53, id 36637)
```

Source of trace:
GIAC Website – http://www.sans.org/y2k/110700.htm
(Stephen Odak)

Detect generated by:
TCPDump

Probability the source address was spoofed:
Very low. It appears to be the Ringzero Trojan, which scans for proxies at ports 80, 8080 and 3128. It needs to receive the response back in order to know the port state on the target, which is impossible if address is spoofed. Also note the fourth packet, if this were a spoofed address and the real host is alive it would have sent a RST back and no retry is sent.

Description of attack:
Packets to ports 80, 8080 and 3128 coming sequentially to one or several hosts.

Attack mechanism:
It seems a compromised box with the RingZero Trojan installed.

Ring Zero is a Trojan virus. It runs as a hidden process on the system target. It sends and retrieves data over an Internet connection. There are three versions of RingZero. This virus scans the network looking for proxies and then sends that information back to a server.
ITS.EXE is one of them; it will copy itself at WINDOWS \SYSTEM directory, this happens when runs ITS.EXE at the first time. Then creates another file RING0.VXD in the same directory. Then, the next time you restart your computer, it creates another file ITS.DAT. In this file it records all the its data. It appears to try to reach two hosts - MEMBERS.ZOOM.COM and PHZFORUM.VIRTUALAVE.NET. The program contains strings that attempt to send mail to an address at PAGER.MIRABILIS.COM through the mail server at WWW.MIRCOSOFT.COM .

Correlation:
http://www.symantec.com/avcenter/venc/dat a/ringzero.trojan.html

http://vil.nai.com/vil/dispVirus.asp?virus_k=10356

Evidence of active targeting:
Assuming this is the RingZero I can be sure this is not active targeting. Accordin g to John Green's paper (The hunt for RingZero) where he explains how the Trojan works: "The pst.exe file appears to be the source of the scanning activity. It does not require an its.dat file to execute, it scans **random** IP addresses, and automatically con solidates successful proxy searches by using the proxy itself"

Severity:
Criticality: 3. No evidence of active targeting.
Lethality: 3. The host could be used to launch attacks through the discovered proxy, or be used in such a way this could cause a DoS.
System countermeasures: 0. Insufficient data, assuming worst case. I can not tell whether the system has a proxy that can be exploited.
Network countermeasures: 5. Because of the retry I could say a packet filtering device is between the IDS (on the outsi de) and the server those packets tried to reach.
Severity = (3 + 3) – (0 + 5) = 1
Defense recommendations:
Defenses seem fine. I would recommend checking firewall ruleset.

Multiple choice question:
Which ports identify the RingZero scan?
a.- 80/tcp, 8080/ tcp, 1080/tcp
b.- 1080/tcp, 80/udp, 27/icmp
c.- 80/tcp, 3128/tcp, 8080/tcp
d.- 80/tcp, 3128/udp, 8080/tcp
Correct answer: C

Detect #2

```
Nov 03 12:56:32 [firewall.ip.address] %PIX -2-106001: Inbound TCP connection denied from
4.17.218.108/3883 to cidr.net .addr.98/21 flags SYN on interface outside
Nov 03 12:56:37 [firewall.ip.address] %PIX -7-106011: Deny inbound (No xlate) tcp src
outside:4.17.218.108/3886 dst outside:cidr.net.addr.101/21
Nov 03 12:56:39 [firewall.ip.address] %PIX -7-106011: Deny inbound ( No xlate) tcp src
outside:4.17.218.108/3887 dst outside:cidr.net.addr.102/21
Nov 03 12:56:40 [firewall.ip.address] %PIX -2-106001: Inbound TCP connection denied from
4.17.218.108/3888 to cidr.net.addr.103/21 flags SYN on interface outside
Nov 03 12:56:42 [firewall.ip.address] %PIX -2-106001: Inbound TCP connection denied from
4.17.218.108/3889 to cidr.net.addr.104/21 flags SYN on interface outside
```

Source of trace:
http://www.sans.org/y2k/111000.htm
(Curt Wilson)

Detect generated by:
Cisco Pix box.

```
Nov 03 12:56:32  [firewall.ip.address]  %PIX-2-106001: Inbound TCP connection denied from
       A                  B                                      C
```

4.17.218.108/3883 to cidr.net.addr.98/21 **flags SYN on interface outside**

```
           D                    E                          F
```

A: Date and timestamp
B: Firewall IP address
C: Informational field
D: Source IP address/source port
E: Destination IP address/destination port
F: Informational field

Probability the address was spoofed:
Very low. This host is probing for FTP serve rs, and it needs to get some kind of answers back in order to make this probe useful.

Description of attack:
This probe is looking for active FTP servers, and then it probably would test it for anonymous access, possibly to put some files to download late r from a compromised machine or for warez distribution. It also would be searching for some vulnerable FTP implementation he could exploit.

Attack mechanism:
It seems an automated scan for this network. Notice that there are 5 connection attempts in only 10 seconds. All of the packets came from the same address, 4.17.218.108 and increasing source port numbers but assuming the PIX firewall drops the packet without notifying the sender we should see retries. I am almost certain that the packets are crafted and then thrown into the network.

Correlation:
This is a very common reconnaisance detect as described in many books and websites.

Evidence of active targeting:
No active targeting, this detect is a scan looking for FTP servers.

Severity:
Criticality: 3 . No evidence of active targeting.
Lethality: 5. There could be a system that is not properly patched and has and FTP server running, this is the worst case.
System countermeasures: 0. For the same stated above, assuming worst case.
Net countermeasures: 5. All attempts have been blocked by the firewall.
(Criticality + Lethality) – (System countermeasures + Net countermeasures) = 3

Defensive recommendations:
Defenses from the outside world seem fine. I would check for latest FTP server patches and irregular user accounts if any server is running.

Multiple choice question:
Which port is usually assigned to FTP servers?
a. 23/udp
b. 21/tcp
c. 25/tcp
d. 110/tcp

Correct answer: b

Detect #3

```
Frame Status Source Address    Dest. Address       Size Rel. Time    Delta Ti     me    Abs. Time            Summary
    1 M      [164.58.70.236]  [00A.00B.00C.1]     60 0:00:00.000 0.000.000     11/06/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
    2        [164.58.70.236]  [00A.00B.00C.3]     60 0   :00:00.041 0.041.103     11/06/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
    3        [164.58.70.236]  [00A.00B.00C.4]     60 0:00:00.063 0.022.058     11/06/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 W  IN=1028
    4        [164.58.70.236]  [00A.00B.00C.8]     60 0:00:00.139 0.076.544     11/06/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
    5        [164.58.70.236]  [00A.00B.00C.14]    60 0:00:00.259 0.119.926     11/0    6/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
    6        [164.58.70.236]  [00A.00B.00C.15]    60 0:00:00.278 0.018.664     11/06/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
    7        [164.58 .70.236]  [00A.00B.00C.16]    60 0:00:00.301 0.022.874     11/06/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
    8        [164.58.70.236]  [00A.00B.00C.21]    60 0:00:00.398 0.097.719     11/06/2000 02:49:28 PM TCP: D=111       S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
    9        [164.58.70.236]  [00A.00B.00C.22]    60 0:00:00.419 0.020.853     11/06/2000 02:49:28 PM TCP: D=111 S=111 SYN FIN
SEQ=1861598755 LEN=0 WIN=1028
   10        [164.58.70.236]  [00A.00B.00C.23]       60 0:00:00.438
```

Source of trace:
http://www.sans.org/y2k/111300.htm
(Luis Mendoza)

Detect generated by:
Unspecified, but fortunately all the fields are explained.

Probability the address was spoof ed:
Very low. This is a scan detect, and if the source address is spoofed, the attacker would never get any response back.

Description of attack:
A group of packets come altogether to the network with increasing host address, but both source port and sequence number fixed. All of the packets have both the SYN and FIN flags set: these packets are certainly crafted.

Attack mechanism:
This probe is searching for active hosts through a poorly configured firewall or IDS, since this is a port that should be ha rdly seen from the outside world. The attacker could be looking specifically for Linux boxes, as they respond to a SYN -FIN with a SYN -FIN-ACK when the port is open or a RST -ACK if the port is closed. He is also looking for any Unix box with its portmapper port open, so he could exploit some RPC vulnerability.

Correlation:

CVE-1999-0168 The portmapper may act as a proxy and redirect service requests from an attacker, making the request appear to co me from the local host, possibly bypassing authentication that would otherwise have taken place. For example, NFS file systems could be mounted through the portmapper despite export restrictions.

CAN-1999-0195 ** CANDIDATE (under review) ** Denial of service in RPC portmapper allows attackers to register or unregister RPC services or spoof RPC services using a spoofed source IP address such as 127.0.0.1.

CAN-1999-0632 ** CANDIDATE (under review) ** The RPC portmapper service is running.

And there are 25 more vulnerabilities in the CVE.MITRE database for RPC services beside these ones.

Evidence of active targeting:
No evidence of activ e targeting, this probe is in the mere phase of reconnaissance.

Severity:
Criticality: 2. No evidence of active targeting
Lethality: 5. Assuming worst case, not much information suppplied.
System countermeasures: 0. Assuming worst case also.
Net countermeasures: 3. This detect could have been picked up by the firewall or by an IDS in the same LAN.
(Criticality + Lethality) – (System countermeasures + Net countermeasures) = 4

Defensive recommendations:
We do not know where this detect was picked up. If it was picked up beyond a firewall I would recommend checking the ruleset. Should RPC be necessary, applying latest patches is a must. If they are not necessary, disable them at all.

Multiple choice question:
Which ports are associated with the portmapper?
a. 111
b. 31337
c. 32771
d. 65535

Correct answers: a and c

<u>Detect #4</u>

```
Nov 13 17:29:18 firewall#1 %PIX -3-106010: Deny inbound udp src outside:200.42.158.62/31338 dst
inside:my.net.64/31337
Nov 13 17:29:18 firewall#1 %PIX -3-106010: Deny inbound udp src outside:200.42 .158.62/31338 dst
inside:my.net.65/31337
Nov 13 17:29:18 firewall#1 %PIX -2-106006: Deny inbound UDP from 200.42.158.62/31338 to
my.net.66/31337
Nov 13 17:29:18 firewall#1 %PIX -2-106006: Deny inbound UDP from 200.42.158.62/31338 to
my.net.67/31337
Nov 13 17:29:18 firewall#1 %PIX -2-106006: Deny inbound UDP from 200.42.158.62/31338 to
my.net.68/31337
Nov 13 17:29:18 firewall#1 %PIX -2-106006: Deny inbound UDP from 200.42.158.62/31338 to
my.net.69/31337
Nov 13 17:29:20 firewall#2 %PIX -2-106006: Deny inbound UDP from 200.42.158.62/31338 to
my.net.223/31337
Nov 13 17:29:20 router#1 240621: Nov 13 20:33:13 UTC: %SEC -6-IPACCESSLOGP: list radius -
out-in denied udp 200.42.158.62(31338) -> my.net.120(31337), 1 packet
Nov 13 17:29:18 router#2 7273583: 7w4d: %SEC -6-IPACCES SLOGP: list arnetmail -out-in denied
udp 200.42.158.62(31338) -> my.net.6(31337), 1 packet
```

(Lines have been cut for brevity, there were over 200 lines)

Source of trace:
My network. All destination addresses and device names have been sanitized.

Detect was generated by:
Two separate Cisco PIX boxes and two separate Cisco routers with the firewall IOS. The fields used for analysis from PIX logs are explained in detect #2.

Description of attack:
The offender scanned an entire class C network looking for a host infected with the BackOrifice trojan. We see logs from four different firewalls because this class C is divided in several subnets.

Attack mechanism:
We see a lot of packets coming to an entire class C network probing for the BackOrifice default port. The BackOrifice client does not have an option nor a plug-in for scanning networks, so we may say that this person is using another tool to scan. It is most probable that he is using a tool such as nmap, where he can set the source port with the –g option. No decoy option has been used to generate any noise, and it is the only scan of this kind in that day, so we could say that this attacker is not a very bright person and has just discovered nmap.

Correlation:
http://www.networkice.com/advice/Phauna/RATs/Back_Orifice/default.htm
CAN-1999-0660 - ** CANDIDATE (under review) ** A hacker utility or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc.
There is also a homepage for this tool, http://www.bo2k.com
This type of probes is very common.

Evidence of active targeting:
None. This is a scan detect, although the attacker is looking for a particular service, he is sending packets to all hosts of a class C network trying to determine if one or several hosts have the 31337/udp port open.

Severity:
Criticality: 3. No evidence of active targeting but this class C network has several sensitive servers, including our main DNS servers.
Severity: 0. All of the hosts in these networks are Unix boxes, and as of now there is no known version of BackOrifice for Unix.
System countermeasures: 5. These systems are not vulnerable to this trojan.
Net countermeasures: 5. All packets have been dropped by our firewalls.
(Criticality + Severity) – (System countermeasures + Net countermeasures) = -2

Defensive recommendations:
Defenses are fine. No packets have reached internal networks. Anyway I would be alert in case a Unix version of this trojan should appear.

Multiple choice question:
Which of the following apply to the BackOrifice trojan?
a. Destination port 31337
b. It is a UDP session
c. Source port number is 31338

Correct answer: None of the above. Listening port can be configured, as well as protocol (UDP or TCP). You cannot configure client source port number.

Evaluation of an attack

Back Orifice 2000

Back Orifice 2000 is a Trojan horse, a hacker infestation tool. It allows remote administration through a TCP/IP network. BO2k gives more control of the remote Windows machine than the person at that machine. It can be found at http://www.bo2k.com . It has been written by The Cult of The Dead Cow group (http://www.cultdeadcow.com) .

In previous versions, infected machines listened at port 31337, but the attacker now configures any port. BO2k, both client and server, run on Windows 95/98/NT and 2000 systems. It can be configured to run over a TCP or UDP, and then the attacker could administrate more then one server at a time. Additionally, it has a very nice GUI which allows newbies to use this tool without prior knowledge (see picture below) .
The attacker can add Plug -ins to improve the use of it. The plug -ins could be in 5 different items: encryption, communication, authentication, server and client enhancement.

With BO2k the attacker gains total control of the system, passwords, registry, etc.
To take control of the remote machine, the attacker has to configure the BO2k file server at his machine, and send it to the victim. He has to configure a network port and a protocol type (UDP or TCP). When the victim executes the file, it copies it self to *$WINDOWS\system\* directory.
Knowing the server's ip address, the protocol and the port, all the attacker has to do is open the BO2k client and click "Connect".

When the client connects to the server, it takes total control over the system.
He can send pings, in this example TCP pings (see trace 1), play sounds repeatedly (see trace 2), transfer files from and to the victim, etc.
At the server, the user does **not know this kind of attack is actually running on his system,** unless the attacker uses som e Plug-ins, such as bo -peep.dll, which allows taking control of the mouse and the keyboard. This software has these capabilities: log keystrokes, copy files, shares directories, kill processes, get lists passwords, list, create and delete keys at the regis try, reboot the system, and a lot of other interesting things.

Formatted

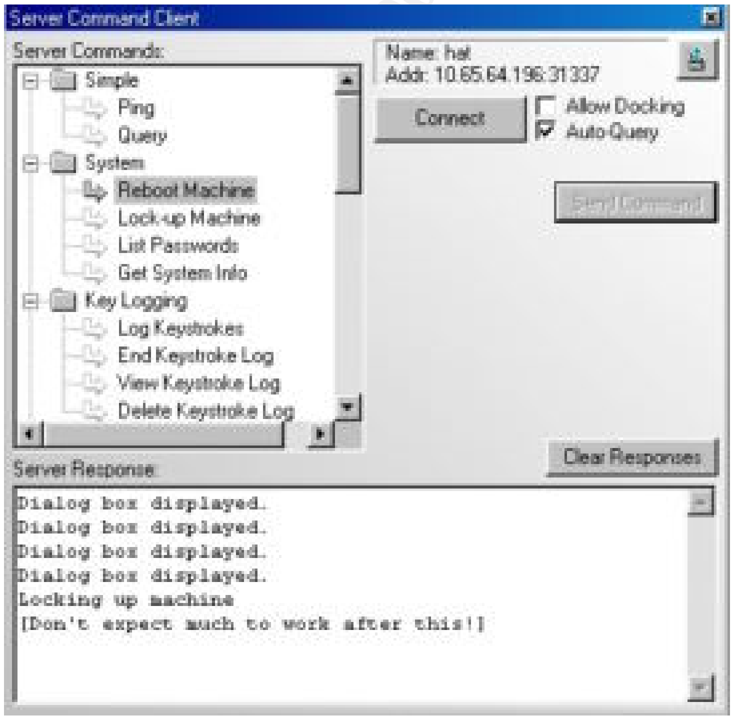
What is most dangerous is that now this application is open source. It means that it will go changing over the time, and probably a signature found today becomes useless tomorrow. We have to w atch for strange traffic to a certain port and then see what is going on in there.

Example 1, TCP pings
```
15:47:33.510110 10.65.64.233.1038 > 10.65.64.196.31337: S 2336637:2336637(0) win 8192 <mss
1460,nop,nop,sackOK> (DF) (ttl 128, id 9989)
15:47:33.510280  10.65.64.196.31337 > 10.65.64.233.1038: S 138517:138517(0) ack 2336638 win
8760 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id 7936)
15:47:33.510984 10.65.64.233.1038 > 10.65.64.196.31337: . ack 1 win 8760 (DF) (ttl 128, id 10245)
15:47:33.511699 10.65.64.23 3.1038 > 10.65.64.196.31337: P 1:139(138) ack 1 win 8760 (DF) (ttl
128, id 10501)
15:47:33.523953 10.65.64.196.31337 > 10.65.64.233.1038: P 1:48(47) ack 139 win 8622 (DF) (ttl
128, id 8192)
15:47:33.527574 10.65.64.233.1038 > 10.65.64.196.31337: P 139:183(  44) ack 48 win 8713 (DF) (ttl
128, id 10757)
15:47:33.548666 10.65.64.196.31337 > 10.65.64.233.1038: P 48:137(89) ack 183 win 8578 (DF) (ttl
128, id 8448)
15:47:33.710696 10.65.64.233.1038 > 10.65.64.196.31337: . ack 137 win 8624 (DF) (ttl 128, id
11013)
```

15:47:33.712670 10.65.64.196.31337 > 10.65.64.233.1038: P 137:1533(1396) ack 183 win 8578 (DF) (ttl 128, id 8704)
15:47:33.910668 10.65.64.233.1038 > 10.65.64.196.31337: . ack 1533 win 8760 (DF) (ttl 128, id 11269).

Example 2, playing sounds

```
15:48:36.74567 5 10.65.64.233.1038 > 10.65.64.196.31337: P 2337273:2337338(65) ack 142720 win 8005 (DF) (ttl 128, id 16133)
15:48:36.824957 10.65.64.196.31337 > 10.65.64.233.1038: P 1:72(71) ack 65 win 8060 (DF) (ttl 128, id 12032)
15:48:36.939105 10.65.64.233.1038 > 10.  65.64.196.31337: . ack 72 win 7934 (DF) (ttl 128, id 16389)
15:48:47.230441 10.65.64.233.1038 > 10.65.64.196.31337: P 65:111(46) ack 72 win 7934 (DF) (ttl 128, id 16645)
15:48:47.253018 10.65.64.196.31337 > 10.65.64.233.1038: P 72:132(60) ack 111 win 8014      (DF) (ttl 128, id 12288)
15:48:47.398756 10.65.64.233.1038 > 10.65.64.196.31337: . ack 132 win 7874 (DF) (ttl 128, id 16901)
15:48:50.602633 10.65.64.233.1038 > 10.65.64.196.31337: P 111:176(65) ack 132 win 7874 (DF) (ttl 128, id 17157)
15:48:50.629714 10. 65.64.196.31337 > 10.65.64.233.1038: P 132:203(71) ack 176 win 7949 (DF) (ttl 128, id 12544)
15:48:50.813689 10.65.64.233.1038 > 10.65.64.196.31337: . ack 203 win 7803 (DF) (ttl 128, id 17413)
15:49:00.092569 10.65.64.233.1038 > 10.65.64.196.31337: P 176:2     22(46) ack 203 win 7803 (DF) (ttl 128, id 17669)
15:49:00.096050 10.65.64.196.31337 > 10.65.64.233.1038: P 203:263(60) ack 222 win 7903 (DF) (ttl 128, id 12800)
15:49:00.263427 10.65.64.233.1038 > 10.65.64.196.31337: . ack 263 win 7743 (DF) (ttl 128, id 17925)
```



Name: hal
Addr: 10.65.64.196:31337

```
Dialog box displayed.
Dialog box displayed.
Dialog box displayed.
Dialog box displayed.
Locking up machine
[Don't expect much to work after this!]
```

"Analyze this" scenario

This is a scenario based question. Your organization has been asked to provide a bid to provide security services for GIAC Enterprises, a dot.com startup that sells electronic fortune cookie sayings. You have been provide d with data a Snort system with a fairly standard rulebase for a month. From time to time, the power has failed, or the disk was full so you do not have data for all days .

For analysis purposes, MY.NET has been replaced with 255.0.

Snort Alert logs star t 11 August 00 and end 14 September 00
Snort Scan logs start 15 August 00 and end 14 September 00
Snort Packet logs start 28 August 00 and end 14 September 00

Signature (click for definition)	# Alerts	# Sources	# Destinations
Possible wu-ftpd exploit - GIAC000623	2	1	2
Happy 99 Virus	2	2	2
site exec - Possible wu-ftpd exploit - GIAC000623	6	1	4
TCP SMTP Source Port traffic	8	2	2
Tiny Fragments - Possible Hostile Activity	12	5	8
External RPC call	40	6	3
Queso fingerprint	54	11	23
Probable NMAP fingerprint attempt	64	7	28
SUNRPC highport access!	64	5	3
NMAP TCP ping!	138	10	42
Null scan!	181	63	73
SMB Name Wildcard	338	17	15
SNMP public access	922	16	1
Attempted Sun RPC high port access	1990	8	11
Watchlist 000220 IL-ISDNNET-990517	5276	19	21
SYN-FIN scan!	5457	6	3005
WinGate 1080 Attempt	6193	347	2156
Watchlist 000222 NET-NCFC	19478	45	19

255.0.1.3, 255.0.1.4 and 200.0.1.5 appear to be the DNS servers, because of the great amount of traffic from a lot of internal hosts to port 53/udp in a normal fashion, although the Snort logs show this as a huge scan toward those machines.

Events of interest:

1. Wingate 1080 Socks

3975 Wingate scans from 347 different sources. Some of them were really big scans and some others are small attemp ts. Two sources made huge scans.

```
09/02-00:20:56.463518  [**] WinGate 1080 Attempt [**] 168.120.16.250:55419 ->
255.0.97.212:1080
09/11-18:40:36.435240  [**] WinGate 1080 Attempt [**] 168.187.26.157:1518 -> 255.0.1.9:1080
```

I recommend sending a "warning" to the owners of these IP addresses, because these are probably compromised hosts conducting scans for Wingate proxies, so intruders can later use them to surf the net, masquerading their real IP addresses.

2. Syn-Fin scans

There were 3065 Syn -Fin alarms, coming from six different sources. Again, two sources sent a huge amount of crafted packets with both SYN and FIN flags set.

They are 210.61.144.125 (starting 9/11 at 6:45 and ending 9/11 at 7:06)
And 213.25.136.60 (starting 9/7 at 21:33 and ending 9/7 at 21:40)

This type of scans penetrates many firewalls since they check only for the SYN flag alone. Once the packet reached the internal host, it could respond with a RST or a SYN-FIN-ACK depending on the box operating system.

```
09/07-21:33:23.187413  [**]  SYN-FIN scan! [**] 213.25.136.60:9704  -> 255.0.1.4:9704
09/07-21:33:23.267012  [**] SYN -FIN scan! [**] 213.25.136.60:9704  -> 255.0.1.8:9704
```

There were also 1524 scan alarms of bogus TCP flags combination apart from the SYN -FIN one. These include FIN -RST, reserved bits or no flag at all, which indicate the presence of malicious activity coming at MY.NET. And one should pay attention to them.

3. Sun RPC high port access attempt

This alert was triggered 1990 times, with 8 different sources. Although six out of these 8 sources come from the same /24 network, they belong to an America Online's /16 address space, so I presume this could be conducted by the same user using a dial-up connection: this means he could get a different IP address each time he connects.
This is an attempt to access portmapper, which also runs on port 32771.

```
09/08-09:57:19.317686  [**] Attempted Sun RPC high port access [**] 205.188.153.112:4000 ->
255.0.105.2:32771
09/08-09:58:19.451527  [**] Attempted Sun RPC high port access [**] 205.188.153.112:4000 ->
255.0.105.2:32771
```

4. SNMP Public access

In the reports generated by Snortsnarf, we see there are 922 alerts, but they are from several **internal** sources to one internal destination. Probably, the technical support installed some software that had to communicate to this host and they forgot to configure the SNMP options.
What is most important from this alert is that if we see connections from internal to internal, the IDS is in the internal network for sure.

```
09/09-15:50:00.735356  [**] SNMP public access [**] 255.0.97.206:1052  -> 255.0.101.192:161
09/09-15:50:01.600870  [**] SNMP public access [**] 255.0.97.206:1053  -> 255.0.101.192:161
```

5. Watchlists

```
09/02-02:43:22.703281  [**] Watchlist 000222 NET   -NCFC [**] 159.226.124.58:2228   ->
255.0.70.33:8765 (China)


09/03-03:37:20.882995  [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.27.111:1526   ->
255.0.206.154:6700
(Israel)
```

I recommend immediate IP blockage, since these two networks are behaving in the same way at least since July. You ca n see this in previous practical submissions.

6. SMB name wildcard

```
09/10-15:29:54.312991  [**] SMB Name Wildcard [**] 255.0.101.160:137     -> 255.0.101.192:137
09/10-15:29:55.817546  [**] SMB Name Wildcard [**] 255.0.101.160:137     -> 255.0.101.192:137
```

This is a powerful gathering technique if the proper measures are not taken. It gives a lot of information about the machine if it runs Windows. It means that the query sent to that host is a request for its NetBios table.

7. Different tools fingerprinting alerts

```
09/02-16:25:42.155404  [**] Probable NMAP fingerprint attempt [**] 63.226.208.41:28518      ->
255.0.253.41:22
09/03-02:17:47.893294  [**] Queso fingerprint [**] 24.19.244.80:6699     -> 255.0.162.200:3889
09/03-11:44:47.774295  [**] Null scan! [**] 129.59.24.21:1540     -> 255.0.204.126:6699
```

These tools are available for free on the Internet and they are very powerful. Nmap can identify which operating system the scanned box is running and which ports are open, giving a brief detail of what they are. Queso can identify which operating system is being scanned by sending bogus TCP flags combinations and examining how they respond to them.

8. **Trolling for Sub7 trojan**

```
Aug 16 04:35:21 35.10.82.111:2814   -> MY.NET.1.6:27374 SYN **S*****
Aug 16 04:35:20 35.10.82.111:2815   -> MY.NET.1.7:27374 SYN **S*****
```

This IP address sent packets to a lot of hosts to port 27374, which is commonly associated with the Sub7 trojan. Where installed, it allows a remote attacker to gain control of the host.

This is the output from ARIN whois:

```
Michigan State University ( NETBLK-MICH-618)
Computer Laboratory
220 Computer Center
East Lansing, MI 48824
US
Netname: MICH-618
Netblock: 35.8.0.0 - 35.10.255.255

Coordinator: Nelson, Doug ( DEN4-ARIN)
nelson@msu.edu
517-353-2980
```

9. Posibble compromised host

255.0.1.13 conducted a huge scan against a big list of internal hosts, particularly for port 7001/udp with source port 7003/udp; although those port could also be in the 70xx range (but not as common). Perhaps a tool was installed on that system to gather information about a particular application that would be later acquired. Anyway, there are no signs of external connections to that host, so it could either be pushing out the gathered information or the attacker could connect on another moment we did not analyze.

```
09/03-09:16:22.105188 [**] spp_portscan: portscan status from 255.0.1.13: 35 connections across 34
hosts: TCP(0), UDP(35) [**]
09/03-09:16:22.283205 [**] spp_portscan: portscan status from 255.0.1.13: 35 connections across 35
hosts: TCP(0), UDP(35) [**]
09/03-09:16:22.800356 [**] spp_portscan: portscan status from 255.0.1.13: 26 connections across 26
hosts: TCP(0), UDP(26) [**]
```

```
Sep  3 09:07:25 255.0.1.13:7003  -> 255.0.153.96:7001 UDP
Sep  3 09:07:25 255.0.1.13:7003  -> 255.0.60.172:7001 UDP
Sep  3 09:07:25 255.0.1.13:7003  -> 255.0.70.134:7001 UDP
Sep  3 09:07:27 255.0.1.13:7003  -> 255.0.53.143:7001 UDP
Sep  3 09:07:27 255.0.1.13:7003  -> 255.0.152.12:7001 UDP
```

Summary

I actually do not know where this IDS is located nor which ruleset file is used for generating these logs. I found some other interesting stuff around those files, such as traffic with source port 0 from and to MY.NET. If I were sure that I see the whole traffic (in and out) I could say that some of that traffic originates from MY.NE T or it is the response to some stimulus. Very hard to determine.

My recommendation is to strengthen perimeter defenses to a maximum, and by doing this try to see which malicious traffic could get into MY.NET. Something useful if you are wondering whether to put an automated response IDS, since with this amount of strange traffic it would be triggering all the time. Also, check the 1.13 host for possible infestation or compromise.

Analysis Process

There were 38 alert and scan log files at http://www.sans.org/NS2000/snort/index.htm
I verified the content of the files to see if there were repeated files, and, oh yeah, SOOS9.txt was the same as SOOS10.txt and SnortA20.txt was the same as SnortA21.t xt. So I eliminated the second files and concatenated all of the SnortA*.txt files into one file called alerts.txt and all of the SnortS*.txt into one file called scans.txt.

I did this in order to process those files with SnortSnarf. This tool produces ht ml files grouping alerts or scans according to type, source or destination address. One of the tables is pasted at the beginning of the analysis. SnortSnarf is available at http://www.silicondefense.com/snortsnarf/index.html
It is very easy to use. It will generate an index page for each file processed, so it is recommendable to put all the files together to have the whole panorama analyzed.

Each alert type has a link that leads you to a statistical page with source and address links.

For example, for "External RPC call" we have a page with this tables:

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total))
18.116.0.75	11	14	1	3
209.160.238.215	10	10	3	3
141.223.124.31	10	10	2	2
161.31.208.237	5	5	1	1
210.100.199.219	3	3	2	2
210.101.101.110	1	3	1	1

Destinations receiving this attack signatu re

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total))
255.0.6.15	35	44	6	9
255.0.100.130	4	10	3	6
255.0.15.127	1	3	1	3

Vi and grep were the other tools I used. Grep to extract lines from log files that contained a specific pattern and vi to see the file, locate that line and see the context.