Global Information Assurance Certification Paper

GIAC Intrusion Detection In Depth

Grégory Lajon

Ottawa Parliament Hill Aug 14-18, 2001

Practical assignment Version 3.0

Assignment 1 – Describe the State of Intrusion Detection

ICMP OS-fingerprinting : The Xprobe challenge.

What is Xprobe ?

In summer 2001, Fyodor Yarochkin and Ofir Arkin released a tool called Xprobe v0.0.1p1 based on the work of Ofir Arkin "ICMP Usage in Scanning". This is a proof of concept tool in its first version. The purpose of this tool is to identify the OS running on a distant host. OS fingerprinting is part of the reconnaissance phase preceding an attack.

This paper is divided in three parts. The first one presents the techniques used by Xprobe. The second shows network traces generated by the tool. The last part discusses some countermeasures necessary to protect the network.

Internet Control Message Protocol (ICMP) is a service protocol in the TCP/IP suite of protocols. There are three categories of ICMP messages : error message, request message and reply message. ICMP is not limited to the ping program. OS fingerprinting based on ICMP is possible due to differences in implementation of ICMP of each operating system.

Xprobe sends UDP and ICMP datagrams to the system being probed. ICMP messages generated by the host in response to the stimuli are then analysed. The results dictates the choice of the next test to apply.

The first stimulus is a UDP datagram sent to a closed port. The expected response is a ICMP port unreachable (type 3 code 3). The payload of an ICMP error message contains the original IP header and at least 64 bits (8 bytes) of original data datagram. The size of data echoed will vary depending on the implementation. Some fields in the original datagram may be wrongly echoed or simply zeroed. In normal usage, only a few fields echoed are actually considered by the receiving host. This is the reason why errors in the following echoed fields usually go unnoticed : IP total length, IP ID, 3 bits Flags, fragment offset, IP header checksum, UDP checksum. Through this analysis, Xprobe is able to categorize OSes.

If the first stimulus did not provide enough information, Xprobe sends a crafted ICMP echo request (type 8). This datagram has a specific TOS value, the DF bit is set and the ICMP code is not null. The victim replies with a ICMP echo reply (type 0). Xprobe watch if the values of TOS, DF and ICMP code are echoed, The TTL value is also analysed. If the results of the previous tests are inconclusive, Xprobe will initiate test for ICMP features not implemented in all IP stacks, i.e. some OSes will not respond to a timestamp request or a netmask request.

In the current version of the tool, a maximum of 4 datagrams will be sent to detect an OS. Xprobe can distinguish several flavours of Windows (95, 98, ME, NT, 2000), a significant improvement in the field of active OS fingerprinting, compared with TCP based OS fingerprinting à la Nmap. This first version of Xprobe was only able to detect a limited amount of OSes and will wrongly identify others. For example, a HP laserjet network printer is detected as a Win2K system.

Xprobe in action :

Usage :

```
#x
X probe ver. 0.0.1p1
------------------
usage: x [-p portnum] [-i interface] [-v] host[/netmask]
#
```

You can choose the destination port number of the initial UDP datagram (default is 32132), the network interface to use and ask for verbose output.

First probe against AIX 4.3.2 on RS/6000

```
# x   xxx.xxx.xxx.78

X probe ver. 0.0.1p1
------------------
Interface: eth0/xxx.xxx.xxx.153

LOG: Target: xxx.xxx.xxx.78
LOG: Netmask: 255.255.255.255
LOG: probing: xxx.xxx.xxx.78
TEST: UDP to xxx.xxx.xxx.78:32132 [98 bytes] sent, waiting for reponse.
TREE: IP total length field value is >20 bytes from the original
TREE: *** AIX!BSDI!NetBSD 1.1.x-1.2.x!MacOS X 1.0-1.2
FINAL:[ AIX ]
```

Xprobe found the correct OS. Let's see tcpdump output of the fingerprinting :

```
15:28:31.352777 xxx.xxx.xxx.153.39451 > xxx.xxx.xxx.78.32132:  udp 70
(DF)
0x0000       4500 0062 bb62 4000 fa11 5ccd xxxx xx99   E..b.b@...\...T.
0x0010       xxxx xx4e 9a1b 7d84 004e 7f57 0000 0000   ...N..}..N.W....
0x0020       0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0030       0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0040       0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0050       0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0060       0000                                      ..

15:28:31.353701 xxx.xxx.xxx.78 > xxx.xxx.xxx.153: icmp: xxx.xxx.xxx.78
udp port 32132 unreachable (DF)
0x0000       4500 0038 dd55 4000 fb01 3a14 xxxx xx4e   E..8.U@...:....N
0x0010       xxxx xx99 0303 e4fa 0000 0000 4500 0076   ..T.........E..v
0x0020       bb62 4000 f611 60cd xxxx xx99 xxxx xx4e   .b@...`...T....N
```

```
0x0030       9a1b 7d84 004e 0000                    ..}..N..
```

As can be seen above with one packet sent and one packet received, it is enough to identify as running AIX.

Xprobe (running on machine xxx.xxxx.xxx.153) sends a UDP packet with a payload of 70 bytes to a closed UDP port on xxx.xxx.xxx.78. The probed machine replies with a ICMP port unreachable (type 3, code 3) and quotes the original IP header + 64 bits of original data datagram (i.e. the UDP header). A look at the quoted header shows that at offset 0x1e (bytes 30-31), the IP datagram length is set to 0x0076. In the original packet it was 0x0062 (seen at offset 2) 0x0076 – 0x0062 = 20. The echoed IP length has been increased by 20, the size of the IP header. A manual computation shows that the echoed IP header checksum does not take into account the error in the IP length field (0x60cd, located at offset 0x26 (bytes 38-39)). According to the white paper, these two characteristics are enough to determine an AIX machine because other systems that increase the IP length header by 20, zeroes the IP header checksum. Also visible is that the UDP checksum has been zeroed (word at offset 0x36 (bytes 54-55))

Xprobe against a NT4 sp5 workstation :

```
# x   xxx.xxx.xxx.80
probe ver. 0.0.1p1
------------------
Interface: eth0/xxx.xxx.xxx.153

LOG: Target: xxx.xxx.xxx.80
LOG: Netmask: 255.255.255.255
LOG: probing: xxx.xxx.xxx.80
TEST: UDP to xxx.xxx.xxx.80:32132 [98 bytes] sent, waiting for reponse.
TREE: IP total length field value is OK
TREE: Frag bits are OK
TEST: ICMP echo request to xxx.xxx.xxx.80 [68 bytes] sent, waiting for
reponse.
TREE: Microsoft Windows Family TCP stack
TREE: Other Windows-based OS (ttl: 127)
TREE: Other Windows-based OS (98/98SE/NTsp3-/NTsp4+)
TEST: ICMP time stamp request to xxx.xxx.xxx.80 [68 bytes] sent, waiting
for reponse.
Receive timeout. Quitting..
TREE: Windows NTsp3-!Windows NTsp4+
TEST: ICMP address mask request to xxx.xxx.xxx.80 [48 bytes] sent,
waiting for reponse.
Receive timeout. Quitting..
FINAL:[ Windows NTsp4+ ]
```

Here again, the correct OS has been found. Let's see the tcpdump output :

```
19:17:12.298303 xxx.xxx.xxx.153.1117 > xxx.xxx.xxx.80.32132:  udp 70
(DF)
0x0000       4500 0062 fbc0 4000 fa11 4b4b xxxx xx99   E..b..@...KK..T.
0x0010       xxxx xx50 045d 7d84 004e 43f2 0000 0000   ..UP.]}..NC.....
```

```
0x0020        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0030        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0040        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0050        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0060        0000                                      ..
19:17:12.301800 xxx.xxx.xxx.80 > xxx.xxx.xxx.153: icmp: xxx.xxx.xxx.80
udp port 32132 unreachable
0x0000        4500 0038 8ca6 0000 7f01 75a0 xxxx xx50   E..8......u...UP
0x0010        xxxx xx99 0303 36db 0000 0000 4500 0062   ..T...6.....E..b
0x0020        fbc0 4000 f911 4c4b xxxx xx99 xxxx xx50   ..@...LK..T...UP
0x0030        045d 7d84 004e 43f2                       .]}..NC.
19:17:12.303412 xxx.xxx.xxx.153 > xxx.xxx.xxx.80: icmp: echo request
(DF) [tos 0x6,ECT]
0x0000        4506 0044 e223 4000 fa01 6510 xxxx xx99   E..D.#@...e...T.
0x0010        xxxx xx50 087b 13ba 8c1a 57b0 0000 0000   ..UP.{....W.....
0x0020        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0030        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0040        0000 0000                                 ....
19:17:12.303957 xxx.xxx.xxx.80 > xxx.xxx.xxx.153: icmp: echo reply (DF)
[tos 0x6,ECT]
0x0000        4506 0044 8da6 4000 7f01 348e xxxx xx50   E..D..@...4...UP
0x0010        xxxx xx99 0000 1c35 8c1a 57b0 0000 0000   ..T....5..W.....
0x0020        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0030        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0040        0000 0000                                 ....
19:17:12.305876 xxx.xxx.xxx.153 > xxx.xxx.xxx.80: icmp: time stamp query
id 32242 seq 37077
0x0000        4500 0044 09bd 0000 fa01 7d7d xxxx xx99   E..D......}}..T.
0x0010        xxxx xx50 0d7b e3bc 7df2 90d5 0000 0000   ..UP.{..}.......
0x0020        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0030        0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0040        0000 0000                                 ....
19:17:22.302686 xxx.xxx.xxx.153 > xxx.xxx.xxx.80: icmp: address mask
request
0x0000        4500 0030 7f19 0000 fa01 0835 xxxx xx99   E..0.......5..T.
0x0010        xxxx xx50 1100 3873 0000 b68c 0000 0000   ..UP..8s........
0x0020        0000 0000 0000 0000 0000 0000 0000 0000   ................
```

Xprobe sent 4 packets and received 2. The ICMP port unreachable payload does not contain errors that can help us. We can notice that the DF bit has not been echoed. Xprobe sends a echo request (ICMP type 8) with a code different from 0 (0x7b, located at offset 0x15 (byte 21)). The TOS byte is set to (0x06). In the reply, the windows host zeroes the ICMP code. This is in contradiction with RFC 792 : “To form an echo reply message, the source and destination addresses are simply reversed, the type code changed to 0, and the checksum recomputed.”. This behavior is a characteristic of windows family of operating systems. The TTL of the echo reply is 0x7f=127. Assuming the initial TTL value is 128 eliminates Windows95, which has an initial TTL value of 32. The TOS field is echoed, so this is not a Win2K host. Now, the host can be NT, 98 or ME. Xprobe sends an ICMP time stamp query which generates no reply therefore the system is Windows NT. Our probed host does not reply to the ICMP address mask request, indicating a Service Pack of 4 or above.

The traces above have been generated on a quiet LAN, with no filtering devices between the hosts. It is very unlikely that datagrams could have been lost.

Countermeasures

How can my IDS detect the use of Xprobe against my network ?

As opposed to nmap, the current version of Xprobe sends only RFC compliant datagrams, making detection difficult without generating false positives.

The first UDP datagram sent to a closed port may look like a port probe. To detect the probes of this particular tool, one can set up a rule on your IDS to detect UDP packets sent to port 32132 with a UDP payload of 70 bytes and a content full of zeros. Since the destination port is adjustable at command line, it is possible to ignore it in the rule. Keep in mind that any UDP datagram of at least 70 bytes will do the job. Detecting variations of the original code may not be possible. A malicious attacker can modify the program, set the source port to 53 and change the payload so that it looks like a DNS reply to a spoofed address. A review of the firewall policies will be necessary in order to limit this reconnaissance methodology.

The echo request packet is easier to detect because it has 2 particularities : TOS=0x06 and ICMP code = 0x7b (123). You may want to detect ICMP echo requests with TOS != 0 and ICMP code != 0 in order to detect variants of the original code.

Depending on the environment, the IDS can rise alerts when unusual ICMP requests like timestamp and netmask are detected at the border.

Firewall countermeasures :

We have seen that ICMP OS fingerprinting attempts detection can be very challenging. Intrusion Detection is good. Intrusion Prevention is better. Maybe it is time to have a closer look at the ICMP traffic you let go through your firewall.

ICMP may be restricted at the firewall without regarding the direction of the traffic. However, we will try to let our internal users being able to ping or traceroute the external world.

Before restricting ICMP traffic, be prepared to explain to your remote users that if ping or traceroute are not working across the firewall, it does not necessarily mean that the firewall or the network is down. If they notice a slow down, it can be due to timeouts because they did not receive error messages like port or host unreachable.

We have seen that outgoing ICMP error messages carry a lot of valuable information that can be used against your organisation. You should block all outgoing error messages, i.e. ICMP type 3, 4, 5, 11 and 12. Be careful with ICMP type 3 code 4 (Fragmentation

needed but DF flag set) if parts of your network have smaller MTU than your Internet link.

Incoming ICMP requests is another category of messages that requires attention. Someone outside your network requesting an address mask or a timestamp may not have the best intents. You can certainly block incoming ICMP type 15,17 and 37 without impacting the users. Blocking incoming echo requests (ping) is a good thing, but may not be possible in your environment. However, check if your filtering device can at least block ICMP type 8 code !=0.

Outgoing ICMP reply corresponding to the blocked incoming ICMP request should also be blocked and an alert raised if detected. We are talking about ICMP type 16, 18 and 38.

Do not forget to limit unneeded incoming UDP traffic if you can.

Conclusion :

Xprobe, even in its beta version is a good proof of concept tool that should help sensibilize security community about ICMP. It is also useful when you want to know more about a machine reported as Windows by nmap. Next versions of Xprobe should be smarter in presence of filtering devices.

References :

Xprobe is released under the GNU general public licence. You can get it at : http://www.sys-security.com/html/projects/X.html

ICMP based remote OS TCP/IP stack finger printing techniques. Ofir Arkin & Fyodor Yarochkin. Phrack 57, file 7 http://www.phrack.org

ICMP usage in scanning V3.0, Ofir Arkin. 2001 http://www.sys-security.com

RFC792 INTERNET CONTROL MESSAGE PROTOCOL by Jon POSTEL, Sept 1981

The Truth About ICMP Lindsay van Eden May 17, 2001 http://www.sans.org/infosecFAQ/threats/ICMP.htm

TCP/IP Illustrated, Volume 1: The Protocols (The Addison-Wesley Professional Computing Series) by W. Richard Stevens

Assignment 2 – Network Detects

Network detect #1 : Nimda spreading attempts

The detect :

```
"19Sep2001" "10:50:42" "log" "drop" "http" "AAA.BBB.175.225" "AAA.BBB.136.192" "tcp"
"19Sep2001" "10:50:42" "log" "drop" "http" "AAA.BBB.175.225" "AAA.BBB.40.184" "tcp"
"19Sep2001" "10:50:42" "log" "drop" "http" "AAA.BBB.175.225" "AAA.BBB.65.128" "tcp"
"19Sep2001" "10:50:42" "log" "drop" "http" "AAA.BBB.175.225" "AAA.BBB.65.198" "tcp"
"19Sep2001" "10:50:42" "log" "drop" "http" "AAA.BBB.175.225" "AAA.BBB.204.61" "tcp"
"19Sep2001" "10:50:42" "log" "drop" "http" "AAA.BBB.175.225" "AAA.BBB.18.184" "tcp"
```

[about 100 similar lines of log per second]

Source of trace :

Network owned and managed by my employer.

Detect was generated by :

Checkpoint FireWall-1. Http traffic was logged in short format. For sake of readability and brevity, the exported log has been trimmed to keep only the interesting fields which are :
“date” “time” “logging action (log or alert)” “FW action (accept, reject or drop)” “destination port (here :http=80/tcp)” “Source IP.” “Destination IP” “protocol”

AAA.BBB.175.225 is an IP address in our network. The destination IP addresses logged are located outside our network. They have the same two first address byte in common.

The pattern was easily detected among thousands of firewall log lines. 115 connections attempts per second originating from the same IP does not go unnoticed, we are not used to have the FW-1 log viewer full of red :-). The firewall was our only tool to detect infected hosts trying to spread. This very high level of activity made the infected hosts easily detected. This detect shows that a firewall analyst who spends time looking at the firewall logs can point out abnormal traffic at the network perimeter.

Probability the source address was spoofed:

Null. The machine is part of our network and has been proved infected by nimda.

Description of the attack:

A Microsoft windows server running Microsoft IIS web server infected by the nimda worm will try to infect other vulnerable IIS servers.

Attack mechanism

A family of vulnerabilities that allows remote capability to execute arbitrary code on a web server with administrator privileges are tried. Several threads are launched simultaneously on the attacking machine. A terrific amount of connections to port 80/tcp are tried mostly on "neighbours", i.e. the first 1, 2 or 3 bytes of the IP address are the same as the infected host. When a connection is successful, 16 different IIS vulnerabilities are tried in order to execute arbitrary code on the server. If this is successful, the remote server is then infected.

Two particularities help the firewall analyst for detecting this :

- very high rate of connections
- destination IP "close" to source IP

If the worm was targeting random IPs at a lower rate, the detect would have been less obvious.

Correlations:

The attack has been detected by a tremendous number of organisations. The threat level reported by incidents.org has been raised to yellow.

Good information on nimda can be found there :
http://www.incidents.org/react/nimda.pdf

Evidence of active targeting:

The worm is targeting every possible web server.

Severity :

(Criticality + Lethality) – (System + Network Counter measures) = severity.

(4+5)-(1+3) = 5

Criticality : The infected machine was used for tests. Unfortunately, it infected our intranet server, used to access critical systems within our organisation : 4
Lethality : Remote administrator access : 5
System counter measures : The web server was not properly patched and is running weak software: 1
Network countermeasures : The Firewall let the attack infect the machine, but drops further infection to the outside. We do not have firewalls inside our network to prevent internal contamination : 3

Defensive recommendation:

If possible, get rid of IIS, as per Gartner's group recommendations
http://www3.gartner.com/DisplayDocument?doc_cd=101034

Web servers accessible from outside should not be part of the internal network. They should be part of the DMZ. In order to contain the infection, some filtering device should prevent web servers to connect to each other within a same network.

Firewalls should block outgoing http traffic originating from web servers.
Firewalls should only allow incoming http traffic to known web servers. This also means that an organisation has to know precisely the servers on its network.

Multiple choice test question:

Which of the following is true ?

a) If the firewall was properly configured, the test server would not have been infected.
b) A firewall can help detecting abnormal network traffic
c) It is not useful to look at firewall logs on a daily basis
d) A firewall can detect ALL malicious network activity.

Answer : b : A firewall can raise alerts when traffic offending the network policy is detected. An analyst who knows the network can also detect suspicious activity, even if is compliant with the network policy. This is not answer a because it is not the firewall's task to block traffic based on content.

Network detect #2 : readme.eml

The detect :

09/27-23:41:07.975574 [**] [1:1284:3] WEB-MISC readme.eml attempt [**] [Classification: Attempted User Privilege Gain] [Priority: 8] {TCP} 161.69.2.149:80 -> 172.28.1.12:1149

Source of trace :

Network owned and managed by my employer.

Detect was generated by :

Snort v1.8 with a set of rules focused on nimda. The rules were posted on http://www.snort.org/article.html?id=31.

Fired rule is :

```
alert tcp $EXTERNAL_NET 80 -> $HOME_NET any \
(msg:"WEB-MISC readme.eml attempt"; \
flags:A+; uricontent:"readme.eml"; nocase; \
classtype:attempted-user; sid:1284; rev:3; \
reference:url,www.cert.org/advisories/CA-2001-26.html;)
```

Probability the source address was spoofed:

Very low, the connection has been established and the web server is transferring the content of the web page to the client located on our network.

Description of the attack:

When nimda infects an IIS server, it will change the content of the web site in order to infect systems running a vulnerable version of Internet Explorer.

Attack mechanism

according to http://www.incidents.org/react/nimda.pdf

"If the worm successfully infects a web server, it uses the HTTP service to propagate itself to clients that browse the web server's pages. Upon infecting a victim server, the worm creates a MIME-encoded copy of itself named "README.EML" and traverses the directory tree searching for web-related files such as those with .HTML, .HTM, or .ASP extensions. Each time the worm finds a web content file, it appends a piece of JavaScript to the file. The JavaScript forces a download of README.EML to any client that views the file via a browser. Some versions of Internet Explorer will automatically execute the README.EML file and allow the worm to infect the client"

Let's have a look at the offending packet (tcpdump –X –n –r snort.log):

```
23:41:07.975574 161.69.2.149.80 > 172.28.1.12.1149: .
409484890:409486350(1460) ack 61608 win 8345 (DF)
0x0000   4500 05dc 0c5d 4000 6f06 a8bc a145 0295        E....]@.o....E..
0x0010   ac1c 010c 0050 047d 1868 3e5a 0000 f0a8        .....P.}.h>Z....
0x0020   5010 2099 1e84 0000 6564 2074 6f20 6578        P.......ed.to.ex
0x0030   6563 7574 6520 7468 6520 7669 7275 732e        ecute.the.virus.
0x0040   203c 423e 5769 6e4e 542f 324b 2073 7973        .<B>WinNT/2K.sys
0x0050   7465 6d73 2063 616e 6e6f 7420 6265 2069        tems.cannot.be.i
0x0060   6e66 6563 7465 6420 6672 6f6d 2061 6e20        nfected.from.an.
0x0070   656d 6169 6c20 6d65 7373 6167 652e 3c2f        email.message.</
0x0080   423e 200d 0a3c 503e 3c2f 503e 0d0a 3c4c        B>...<P></P>..<L
0x0090   493e 5768 656e 2069 6e66 6563 7469 6e67        I>When.infecting
0x00a0   2c20 6974 2061 7070 656e 6473 202e 4153        ,.it.appends..AS
0x00b0   502c 202e 4854 4d2c 2061 6e64 202e 4854        P,..HTM,.and..HT
0x00c0   4d4c 2064 6f63 756d 656e 7473 2c20 616e        ML.documents,.an
0x00d0   6420 6669 6c65 7320 6e61 6d65 6420 494e        d.files.named.IN
0x00e0   4445 582c 204d 4149 4e2c 2061 6e64 2044        DEX,.MAIN,.and.D
```

```
0x00f0   4546 4155 4c54 2c20 7769 7468 206a 6176        EFAULT,.with.jav
0x0100   6173 6372 6970 7420 636f 6465 2077 6869        ascript.code.whi
0x0110   6368 2063 6f6e 7461 696e 7320 696e 7374        ch.contains.inst
0x0120   7275 6374 696f 6e73 2074 6f20 6f70 656e        ructions.to.open
0x0130   2061 206e 6577 2062 726f 7773 6572 2077        .a.new.browser.w
0x0140   696e 646f 7720 636f 6e74 6169 6e69 6e67        indow.containing
0x0150   2074 6865 2069 6e66 6563 7469 6f75 7320        .the.infectious.
0x0160   656d 6169 6c20 6d65 7373 6167 6520 6974        email.message.it
0x0170   7365 6c66 2028 7461 6b65 6e20 6672 6f6d        self.(taken.from
0x0180   2074 6865 2064 726f 7070 6564 2066 696c        .the.dropped.fil
0x0190   6520 5245 4144 4d45 2e45 4d4c 292e 2054        e.README.EML)..T
0x01a0   6875 7320 7768 656e 2074 6869 7320 696e        hus.when.this.in
0x01b0   6665 6374 6564 2077 6562 2070 6167 6520        fected.web.page.
0x01c0   6973 2061 6363 6573 7365 6420 286c 6f63        is.accessed.(loc
[...]
```

Well, it looks like a false positive. This is not javascript code. The user was getting information on nimda at NAI.com. The rule triggered when it detected the string "readme.eml".

Correlations:

Nimda had a high impact on the Internet community. Many organisation have seen this attack.

http://www.incidents.org/react/nimda.pdf

Adjust your IDS rules to limit false positives. A replacement rule has been proposed by "eyes to the Skies" referenced at :
http://www.incidents.org/archives/intrusions/msg01818.html

alert TCP $EXTERNAL_NET 80 -> any any (msg: "nimda-infection-attempt_html, incoming"; flags: A+; content: "window.open\(\"readme.eml";)

Severity :

(Criticality + Lethality) – (System + Network Counter measures) = severity.

(2+0)-(3+3)=-4

Criticality : targeted system is a workstation : 2
Lethality : false positive : 0
System countermeasure : We do not know which browser is used : 3
Network countermeasure : There is an IDS, but nothing blocks the infection at the network level : 3

Defensive recommendation:

Use a non vulnerable browser or be sure to have the latest patches applied. Disable javascript. Make sure antivirus software have the latest signature file.

Multiple choice test question:

Which command displays the content of the packet ?

a) tcpdump –r snort.log –n
b) tcpdump –r snort.log –w snort.ip.log icmp
c) tcpdump –r snort.log –n –X
d) cat alert

answer : c:

Network detect #3 : Web content filter evasion.

The detect :

My boss over the phone : "Hey, Greg ! There is something strange with the superscout logs. Someone is going to hacking sites and superscout says it allows it ! I am sure I have not defined an exception for this IP"

Source of trace :

Network owned and managed by my employer. We have policies concerning the usage of Internet. High level management decided that "hacking" web sites should be blocked.

Detect was generated by :

Superscout is a web content filter edited by surfcontrol. The software runs on a computer located on our Internet link. The network interface is in promiscuous mode, i.e. sniffer mode. Every web request detected is compared with its database of URLs. If the request is not compliant with our policies, superscout spoofs TCP reset packets to the server and spoofs the response to the client. The client receives a web page explaining the policy. If the web content filter crashes, web access is still possible.

http://www.surfcontrol.be/en/superscoutweb.html

A linux box running tcpdump has been installed on the network to capture web traffic.

Probability the source address was spoofed:

Low. The source is inside our network and the TCP connection has been established. The user who tries to evade the web content filter does not want to be caught. He may have stolen any unused IP address from the same network to hide his identity.

Description of the attack:

Splitting a request into several small packets to elude detection from a web content filtering device.

Attack mechanism

here is the attempt from my pc to reach www.2600.com :

19:15:19.217885 X.X.X.X.2124 > 207.99.30.230.80: S 693376240:693376240(0) win **5840** <mss **1460**,sackOK,timestamp 1208677 0,nop,wscale 0> (DF)
19:15:19.233547 207.99.30.230.80 > X.X.X.X.2124: S 1266803559:1266803559(0) ack 693376241 win **17376** <mss 1460,nop,wscale 0,nop,nop,timestamp 5871585 1208677> (DF)
19:15:19.233679 X.X.X.X.2124 > 207.99.30.230.80: . ack 1 win 5840 <nop,nop,timestamp 1208678 5871585> (DF)
19:15:19.235409 X.X.X.X.2124 > 207.99.30.230.80: P 1:253(252) ack 1 win 5840 <nop,nop,timestamp 1208678 5871585> (DF) [this is the HTTP GET request]
19:15:19.237259 207.99.30.230.80 > X.X.X.X.2124: FP 1:994(993) ack 253 win **5840** (DF) [this is the spoofed answer]
19:15:19.237300 207.99.30.230.80 > X.X.X.X.2124: R 1266803560:1266803560(0) win 5840 (DF)
19:15:19.237345 X.X.X.X.2124 > 207.99.30.230.80: . ack 995 win 6951 <nop,nop,timestamp 1208679 5871585> (DF)
19:15:19.240630 X.X.X.X.2124 > 207.99.30.230.80: F 253:253(0) ack 995 win 6951 <nop,nop,timestamp 1208679 5871585> (DF)
19:15:19.252867 207.99.30.230.80 > X.X.X.X.2124: R 1266804554:1266804554(0) win 16384
19:15:19.255732 207.99.30.230.80 > X.X.X.X.2124: R 1266804554:1266804554(0) win 16384

After the 3 way handshake, my HTTP GET request is sent in one packet. Then I get the spoofed response (the TCP window size of the "web site" dropped from 17376 to 5840), this is line 5. I also receive extraneous resets before the connection is closed.

Now, let's have a look at the other machine :

19:17:30.476231 Y.Y.Y.Y.1027 > 207.99.30.230.80: S 828714826:828714826(0) win 1024 <mss **256**,sackOK,timestamp 1221802 0,nop,wscale 0> (DF)
19:17:30.493089 207.99.30.230.80 > Y.Y.Y.Y.1027: S 1297815882:1297815882(0) ack 828714827 win 16592 <mss 1460,nop,wscale 0,nop,nop,timestamp 5871847 1221802> (DF)
19:17:30.493236 Y.Y.Y.Y.1027 > 207.99.30.230.80: . ack 1 win 1024 <nop,nop,timestamp 1221803 5871847> (DF)
19:17:30.495033 Y.Y.Y.Y.1027 > 207.99.30.230.80: . 1:199(198) ack 1 win 1024 <nop,nop,timestamp 1221803 5871847> (DF) [this is the first part of the HTTP GET]
19:17:30.495106 Y.Y.Y.Y.1027 > 207.99.30.230.80: P 199:253(54) ack 1 win 1024 <nop,nop,timestamp 1221803 5871847> (DF) [this is the second part of the HTTP GET]
19:17:30.520798 207.99.30.230.80 > Y.Y.Y.Y.1027: . 1:245(244) ack 253 win 16592 <nop,nop,timestamp 5871848 1221803> (DF)
19:17:30.520844 Y.Y.Y.Y.1027 > 207.99.30.230.80: . ack 245 win 1464 <nop,nop,timestamp 1221806 5871848> (DF)
[...]
19:17:31.602056 207.99.30.230.80 > Y.Y.Y.Y.1027: . 27773:28017(244) ack 253 win 16592 <nop,nop,timestamp 5871850 1221907> (DF)

19:17:31.602099 Y.Y.Y.Y.1027 > 207.99.30.230.80: . ack 28017 win 8296 <nop,nop,timestamp 1221914 5871850> (DF)
19:17:31.603226 207.99.30.230.80 > Y.Y.Y.Y.1027: FP 28017:28197(180) ack 253 win 16592 <nop,nop,timestamp 5871850 1221908> (DF)
19:17:31.604538 Y.Y.Y.Y.1027 > 207.99.30.230.80: F 253:253(0) ack 28198 win 8296 <nop,nop,timestamp 1221914 5871850> (DF)
19:17:31.631105 207.99.30.230.80 > Y.Y.Y.Y.1027: . ack 254 win 16592 <nop,nop,timestamp 5871850 1221914> (DF)

The MSS (Maximum Segment Size) has been set to 256 on the first line. This is small. The HTTP GET request is split in two different packets (lines 4 and 5). If the web content filter does not do some kind of TCP stream reassembly, it can be bypassed. Note that a small MSS generates a lot of overhead, because a large number of packets are carrying a small payload.

Actually, we found out that the MTU on the PC was set to 250 (instead of 1500 for standard Ethernet).

Correlations:

Neil Desai posted the following on Bugtraq :

```
> I have been working with the people of SurfControl for
> a couple of weeks now and all they say is that they
> will submit it as a bug in the software and try to get
> a fix out in the next couple of months. So here goes?.
> You can bypass the software by using a proxy sever
> before your traffic is looked at by SurfControl Super
> Scout. After talking with the people at SurfControl it
> has become apparent that you may bypass all of their
> software that is meant for Internet monitoring. I have
> not been able to test it though. They only look at
> packets that have the HTTP GET request and "Host:"
> information in it. If you split up the request so that
> HTTP GET request is not in the same packet as
> the "Host:" information then you will bypass the
> software.
> You can easily do this by using a proxy server before
> you get to the node that is doing the Internet
> monitoring. If you have Compaq PC's or servers that
> are not patched you can proxy off the Insite Manager
> software
> (http://www.compaq.com/support/files/server/us/dow
> nload/9609.html). If you have PERL installed you can
> use RFProxy, HTTPush or Pudding. These programs
> were intended for the testing of IDS evasion
> techniques but work wonders for Internet
> monitoring/blocking evasion.
```

The message ID is : 20010618234934.27318.qmail@securityfocus.com, it was posted on June 18th 2001.

Severity :

In this case, the severity formula is not easy to use. The "victim" was not a computer system.

(Criticality + Lethality) – (System + Network Counter measures) = severity.

(0+3)-(3+0) = 0

Criticality : No critical systems involved : 0
Lethality : Inappropriate use of the network infrastructure : 3
System countermeasures : Very few of our users know how to change the MTU on their machine : 3
Network Countermeasures : The web content filter fails with small TCP segments. We have to wait for a patch and then apply the patch. : 0

Defensive recommendation:

The management can recall users about the appropriate use of the network infrastructure. They should explain why blocking hacking sites is important for our business.

Until we get the patch, we may try to set up a filter that detects low MSS. I am not sure the threat worth the work.

Multiple choice test question:

Reducing the MTU on your system

a) increase your network bandwidth
b) prevent you to communicate with computers in Russia
c) can make your network traffic undetectable by an IDS
d) protects your computer against buffer overflows.

Answer : c. By reducing the maximum size of an IP packet, an offending command may be split in separate IP packets. If the IDS is not able to reassemble the command, the full pattern will not match and the attack will go unnoticed.

network detect #4 : SYN-FLOOD ?

The detect :

"13Oct2000" "16:36:31" "en1" "log" "accept" "smtp" "211.217.224.186" "VICTIM" "tcp" "2918" " len 48"

"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "164.100.10.18" "VICTIM" "tcp" "65020" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "211.202.2.125" "VICTIM" "tcp" "1716" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "211.38.248.1" "VICTIM" "tcp" "55427" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "195.166.231.4" "VICTIM" "tcp" "4406" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "203.2.192.109" "VICTIM" "tcp" "44988" " len 48"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "210.190.22.194" "VICTIM" "tcp" "1858" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "213.29.83.2" "VICTIM" "tcp" "64913" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "211.118.129.2" "VICTIM" "tcp" "3096" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "194.204.145.21" "VICTIM" "tcp" "2302" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "211.101.132.136" "VICTIM" "tcp" "43192" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "203.228.233.9" "VICTIM" "tcp" "64656" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "217.52.34.2" "VICTIM" "tcp" "3815" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "211.250.6.194" "VICTIM" "tcp" "2430" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "211.23.74.82" "VICTIM" "tcp" "1887" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "211.94.65.199" "VICTIM" "tcp" "53797" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "209.119.134.2" "VICTIM" "tcp" "2138" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "202.168.255.42" "VICTIM" "tcp" "2159" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "200.51.40.230" "VICTIM" "tcp" "2443" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "210.109.185.1" "VICTIM" "tcp" "3478" " len 48"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "203.252.201.3" "VICTIM" "tcp" "39543" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "202.106.127.124" "VICTIM" "tcp" "3662" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "193.231.215.201" "VICTIM" "tcp" "1694" " len 60"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "203.229.144.250" "VICTIM" "tcp" "64513" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "207.67.131.32" "VICTIM" "tcp" "2875" " len 48"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "202.184.170.51" "VICTIM" "tcp" "42095" " len 44"
"13Oct2000" "16:36:31" “en1” “log” "accept" "smtp" "203.80.255.25" "VICTIM" "tcp" "1444" " len 44"

[about 30 similar logs per second]

Source of trace :

Perimeter of the network owned and managed by my employer. VICTIM is a machine in our network.

Detect was generated by :

Checkpoint FireWall-1. The exported log has been trimmed. Empty fields have been removed, as well as non interesting fields. The log format is as follows :

“date” “time” “FW interface” “logging action : log or alert” “FW action : accept, reject or drop” “destination port : here smtp :25/tcp” “source” “destination” “protocol” “source port” “information”

This pattern has been discovered when analysing logs of a newly installed firewall. I was trying to find internal mail servers that the company would not know about. So the filters were : incoming connections (FW interface =en1), service = smtp and destination address different from known mail servers. I have been surprised to discover such an important traffic hidden in the logs. I tried to get more information about this “mail server”. In the IP database of the company, this was registered as being a windows 95 workstation. Several “telnet VICTIM 25” at different times in the day showed that port 25 was closed. This workstation was the only machine in our network to be targeted.

Probability the source address was spoofed:

At first glance, this log looks like a SYN flood attack. For this kind of denial of service attack, usually source addresses are spoofed. Unfortunately, the firewall logs do not show the packet. We can not analyse the TTL, IP ID and other fields for indication of crafted packets. A SYN flood on a closed port is not very efficient. A test on a random sample of source addresses showed that 95% of them are alive two hours after the detect and have port 25 opened. This is certainly not an efficient way of SYN flooding.

The connection attempts have been running for months.

This may not be a SYN flood attack.

If it is not a SYN flood attack and source addresses are not spoofed, we have to find out why so many real mail servers across the world want to send mail to a workstation.

Description of the attack:

Lots of connection attempts from very different sources to port 25 on a workstation.

Attack mechanism

The next step is to find out why this particular workstation is targeted. A DNS reverse lookup shows a particularity in the name. It resolves as IS.subdomain.mycompany.com. The other workstations usually resolve as wsxxxyyy.subdomain.mycompany.com.

Correlations:

A phone call to the mail servers guy informed me that IS.subdomain.mycompany.com was an old mail server, removed 2 years ago from the network. The DNS entry has never been removed and the IP has been reused. The user never noticed that her machine was receiving an average of 30 connections attempts per second.

Severity :

(Criticality + Lethality) – (System + Network Counter measures) = severity.

(2+2)-(5+3)=-4

Criticality : this is a standard workstation : 2
Lethality : No break-in attempts, but some bandwidth used and FW logs filled : 2
System : Non vulnerable to the attack : 5
Network countermeasures : A firewall was freshly installed and logged the traffic but the ruleset was too permissive : 3

Defensive recommendation:

Make sure to remove specific DNS entries when a server is removed and the IP address is reused.

Know your network and allow mail traffic only from and to your mail servers. If people inside your network are using external mail server, you have to open outgoing smtp to these mail servers.

Multiple choice test question:

Which of the following information can you get from Firewall-1 logs ?

a) TTL and IP ID of the packet
b) The payload
c) Remote operating system
d) Direction of an attempted connection : Incoming or outgoing ?

Answer : d FireWall-1 logs connections attempts. The firewall interface logged allows you to know where the packet came from. Another way to know is to look at source and destination. But if your network has 40 different class C, you may want to have an easier way to find it.

Network detect #5

The detect :

16:24:27" src=144.122.72.200 dst=x.x.8.199 src_port=10 dst_port=10 service=tcp/port:10 proto=6 policy_id=44 direction=incoming duration=59 sent=64 rcvd=0 action=Permit 1
16:24:27" src=144.122.72.200 dst=x.x.8.200 src_port=10 dst_port=10 service=tcp/port:10 proto=6 policy_id=44 direction=incoming duration=59 sent=64 rcvd=0 action=Permit 1
16:24:27" src=144.122.72.200 dst=x.x.8.201 src_port=10 dst_port=10 service=tcp/port:10 proto=6 policy_id=44 direction=incoming duration=59 sent=64 rcvd=0 action=Permit 1
16:24:27" src=144.122.72.200 dst=x.x.8.202 src_port=10 dst_port=10 service=tcp/port:10 proto=6 policy_id=44 direction=incoming duration=59 sent=64 rcvd=0 action=Permit

Source of trace :

Incidents.org archive page
post from Mary M. Chaddock on Wed, 11 Jul 2001
Subject: Port scans: src port 10 -> dst port 10

"I noticed an unusual scan in yesterdays log (small part of it listed below). Any ideas what they may be looking for on TCP 10?"

http://www.incidents.org/archives/intrusions/msg01034.html

Detect was generated by :

It was not mentioned in the post. However, it looks like some Firewall log.

Probability the source address was spoofed:

A scan is a reconnaissance technique. To get the information back, the address should not be spoofed. An attacker may spoof a source address of a scan in order to keep security staff busy.

Description of the attack:

Network scan on port 10/TCP with reflexive ports.

Attack mechanism

The attacker sweeps the network range with connections attempts to port 10/tcp. Port 10/tcp is not a well known port. A deep search on the Internet did not reveal any known program using this port. I see three possibilities here :
1 the attacker was looking for closed tcp port for reconnaissance
2 the attacker was playing with a new scanner. He chose 10 as a port because the scanner was asking for a number and he did not know what to choose.
3 this is a new trojan/backdoor using port 10.

Except with NetBios and domain, it is unusual to see source port equals to destination port.

Correlations:

This is a rarely probed port. A search with google failed to find some correlation. Port scans are part of the Internet.

Severity :

(Criticality + Lethality) – (System + Network Counter measures) = severity.
(3+2)-(2+3)=0

Criticality : I do not know the network. : 3
Lethality : Scan on closed port : 2
System : I do not know the systems : 2
Network counter measure : The firewall let incoming 10/tcp. but the scan has been detected : 3

Defensive recommendation:

Block unused ports at the firewall. Check your logs to see if a machine on the network replied with a SYN-ACK. If you do not have the information in your logs, you may want

to scan your network on port 10. It can be the new trojan of the month. Check if other people had the same type of scan.

Multiple choice test question:

Which of the following is wrong ?

Network scans

a) shows current trends in vulnerabilities
b) shows nothing interesting, it is just script kiddies playing
c) are part of the Internet activity
d) are part of the reconnaissance phase, preceding an attack

answer : b : It is important to know the ports that are often scanned and make sure the network is not running vulnerable software on these ports.

Assignment 3 – "Analyse This !" Scenario

The following report summarises an extended analysis of several Snort logs captured on a university network. The logs were captured during 5 days, from September 1st to September 5th ,2001. A fair amount of unwanted or potentially malicious activity has been detected.

The data has been provided in 3 sets of files : alert.01090[1-5], oos.Sep.[1-5].2001 and scans.01090[1-5]. Format of files will be described in the analysis process part.

Forewords : the alert logs have been provided without the set of rules or the location of sensors on the network. We have no knowledge of the network topology.

Summary :

5 days during which :

- 135 different alert types have been logged in the alert file.
- 780,000 alerts raised
- 383 packets out of spec
- 36550 hosts on MY.NET network involved.

top 10 alerts :

Rank	number of alerts	Alert	% of total
1	305468	WEB-MISC Attempt to execute cmd	39%
2	268112	IDS552/web-iis_IIS ISAPI Overflow ida nosize	34%
3	32311	ICMP Destination Unreachable (Communication Administratively Prohibited)	4%
4	20678	MISC Large UDP Packet	3%
5	20453	MISC traceroute	3%
6	19590	MISC source port 53 to <1024	3%
7	15458	CS WEBSERVER – external web traffic	2%
8	14853	INFO MSN IM Chat data	2%
9	12258	WEB-MISC prefix-get //	2%
10	10805	ICMP Echo Request Nmap or HPING2	2%

We have 79873 distinct sources of alerts including 2046 internal hosts.

The 2 first alerts have been generated by 66769 distinct source IPs (83% of total distinct source IPs). The other alerts have been generated by 13237 different sources.

The two first alerts are now part of the background noise on the Internet. In order to have more accurate analysis, the 2 top alerts will be removed for the rest of the process analysis.

top 10 talkers for alerts:

IP	Alerts	Comments
MY.NET.14.1	16091 ICMP Destination Unreachable (Communication Administratively Prohibited) 531 ICMP traceroute	see Alert Analysis #3
MY.NET.16.5	14700 ICMP Destination Unreachable (Communication Administratively Prohibited)	see Alert Analysis #3
61.153.17.244	8898 MISC Large UDP Packet	see Alert Analysis #4
61.153.17.24	6652 MISC Large UDP Packet	see Alert Analysis #4
MY.NET.226.18	5302 ICMP Echo Request Nmap or HPING2	only 2 destinations 204.71.200.75 and 206.79.171.51
MY.NET.30.2	4690 ICMP Destination Unreachable (Network Unreachable)	tried by 1172 different hosts
130.161.37.101	4242 High port 65535 tcp - possible Red Worm - traffic	see scanner analysis #3
MY.NET.98.190	3853 Possible trojan server activity	network scan port 27374 (subseven)
208.26.55.145	3333 Tiny Fragments - Possible Hostile Activity 70 Null scan!	See Alert Analysis #5
MY.NET.208.82	3393 ICMP Echo Request Nmap or HPING2	4 different destinations.

ALERT ANALYSIS 1 & 2

1st alert : WEB-MISC Attempt to execute cmd
2nd alert : IDS552/web-iis_IIS ISAPI Overflow ida nosize

Categorisation : attack without preceding reconnaissance.

Comments :

These alerts are raised by a family of worms that uses vulnerabilities in Microsoft IIS web server. This can be DOS.Storm, code red, code blue. Sept 1st to Sept 5th is too early

for nimda. Some targeted attacks may have occurred, but they have been lost in the flood of worms attacks.
34841 different machines hit in 5 days, with an average of 9 hits per machine. All the attacks came from outside. I do not know if the IDS was set to detect attacks coming from internal network. On MY.NET network, 34841 machine have port 80/tcp opened, this is quite a lot. Depending on site's policies, this may require further investigation.

Links :

http://www.incidents.org/react/nimda.pdf

ALERT ANALYSIS #3

ICMP Destination Unreachable (Communication Administratively Prohibited)

Categorisation : Denied access.

Comments :

This kind of message can be sent by a router when it receives a packet that violates its control lists.

2 major sources : MY.NET.14.1 and MY.NET.16.5

All the ICMP messages generated by MY.NET.14.1 and MY.NET.16.5 have been directed to hosts in MY.NET . We can imagine that a part of the University Network is restricted from the rest and a sensor is between the two networks. We may want to investigate why 427 different hosts wanted to reached MY.NET.14.1. It can be a configuration error. It may also be students who are trying to change their marks on the administration network :-).

Sample :

09/02-07:00:30.052619 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.16.5 -> MY.NET.110.90
09/02-07:00:42.136216 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.14.1 -> MY.NET.219.154
09/02-07:00:45.139521 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.16.5 -> MY.NET.219.154
09/02-07:00:48.588473 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.16.5 -> MY.NET.202.238
09/02-07:00:50.054502 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.14.1 -> MY.NET.110.88

09/02-07:00:58.152150 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.16.5 -> MY.NET.5.74
09/02-07:01:05.340346 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.16.5 -> MY.NET.210.98
09/02-07:01:08.417633 [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.16.5 -> MY.NET.204.214

ALERT ANALYSIS #4

20678 " MISC Large UDP Packet"

2 important sources of alerts:
61.153.17.244 has sent 8898 MISC Large UDP Packet
61.153.17.24 has sent 6752 MISC Large UDP Packet

Categorisation : Suspicious traffic.

Comments :

3 machines are targeted : MY.NET.111.142, MY.NET.144.51, MY.NET.111.221. 61.153.17.24 is only targeting MY.NET.111.221. We see a large amount of traffic coming from port 0 to port 0. It is interesting to notice that 61 different IPs have sent large UDP Packet with source port = 0 and dest port=0. This is usually a bad sign, it means that the packet is crafted. The two IP addresses belong to the same owner. We see activity from both addresses at the same time.

09/05-19:03:21.044684 [**] MISC Large UDP Packet [**] 61.153.17.244:0 -> MY.NET.111.142:0
09/05-19:03:21.179799 [**] MISC Large UDP Packet [**] 61.153.17.24:0 -> MY.NET.111.221:0
09/05-19:03:21.276811 [**] MISC Large UDP Packet [**] 61.153.17.24:0 -> MY.NET.111.221:0
09/05-19:03:21.952701 [**] MISC Large UDP Packet [**] 61.153.17.244:0 -> MY.NET.111.142:0
09/05-19:03:23.046385 [**] MISC Large UDP Packet [**] 61.153.17.244:30699 -> MY.NET.111.142:21775
09/05-19:03:23.357940 [**] MISC Large UDP Packet [**] 61.153.17.244:0 -> MY.NET.111.142:0
09/05-19:03:24.278128 [**] MISC Large UDP Packet [**] 61.153.17.24:3563 -> MY.NET.111.221:1548
09/05-19:03:24.451546 [**] MISC Large UDP Packet [**] 61.153.17.244:3563 -> MY.NET.111.142:1548
09/05-19:03:24.676639 [**] MISC Large UDP Packet [**] 61.153.17.24:3563 -> MY.NET.111.221:1548
09/05-19:03:26.061992 [**] MISC Large UDP Packet [**] 61.153.17.244:0 -> MY.NET.111.142:0
09/05-19:03:26.247438 [**] MISC Large UDP Packet [**] 61.153.17.244:0 -> MY.NET.111.142:0
09/05-19:03:26.379935 [**] MISC Large UDP Packet [**] 61.153.17.24:53120 -> MY.NET.111.221:16561

The packets are not sent quick enough to constitute a DOS attack. It is rare when the rate of 2 packets per second is surpassed. Therefore, the attacker has nothing to gain with a spoofed source IP.

link

According to Matt Scarborough , in an article on Network Gaming
http://www.incidents.org/detect/gaming.php

"*"On some proxy servers, such as Microsoft Proxy Server, you will need to open UDP port 0 as an additional Subsequent UDP Inbound port." (Q236430)
http://support.microsoft.com/support/Games/Zone/FAQ/connect.asp
"

In our case, the UDP ports used are not well known gaming ports, but it may be a new game.

registration information

```
whois -h whois.apnic.net 61.153.17.244

% Rights restricted by copyright. See
http://www.apnic.net/db/dbcopyright.html
% (whois6.apnic.net)

inetnum: 61.153.17.0 - 61.153.17.255
netname: NINGBO-ZHILAN-NET
descr: NINGBO TELECOMMUNICATION CORPORATION ,ZHILAN APPLICATION SERVICE
PROVIDER
descr: Ningbo, Zhejiang Province
country: CN
admin-c: CZ61-AP
tech-c: CZ61-AP
mnt-by: MAINT-CHINANET-ZJ
changed: master@dcb.hz.zj.cn 20010512
source: APNIC

person: CHINANET ZJMASTER
address: no 378,yan an road,hangzhou,zhejiang
country: CN
phone: +86-571-7015441
fax-no: +86-571-7027816
e-mail: master@dcb.hz.zj.cn
nic-hdl: CZ61-AP
mnt-by: MAINT-CHINANET-ZJ
changed: master@dcb.hz.zj.cn 20001219
source: APNIC
```

ALERT ANALYSIS #5

208.26.55.145
  3333 Tiny Fragments - Possible Hostile Activity
   70   Null scan!

Categorisation : reconnaissance/attack.

Comments:

Only one host targeted in the network : MY.NET.178.236 .

During 83s on Sept 02 06:24:10 to 06:25:33 here is what has been detected :

3333 tiny fragments (40 frags/s)
70 null scans
1291 Syn packets targeting 882 different ports <=1024 + one attempt on 2048
5 OOS packets logged in the scan log.

It is hostile traffic.

I do not know what is the intent of the fragmented traffic. It would be interesting to analyse the content of the packets. If the intent was to elude IDS, why doing a noisy scan at the same time. If it was in a DOS intent, why doing a port scan at the same time ?

Nothing has been reported in the OOS log. This leads questions regarding the efficiency or location of the OOS sensor.

For information : according to IIS attacks logs, MY.NET.178.236 has port 80/tcp opened.

Scan :

```
[...]
Sep  2 06:24:35 208.26.55.145:42211 -> MY.NET.178.236:700 SYN ******S*
Sep  2 06:24:35 208.26.55.145:42211 -> MY.NET.178.236:705 SYN ******S*
Sep  2 06:24:35 208.26.55.145:42211 -> MY.NET.178.236:898 SYN ******S*
Sep  2 06:24:37 208.26.55.145:46812 -> MY.NET.178.236:0 NULL ********
Sep  2 06:24:35 208.26.55.145:42210 -> MY.NET.178.236:979 SYN ******S*
Sep  2 06:24:35 208.26.55.145:42210 -> MY.NET.178.236:97 SYN ******S*
Sep  2 06:24:36 208.26.55.145:42211 -> MY.NET.178.236:216 SYN ******S*
Sep  2 06:24:36 208.26.55.145:42210 -> MY.NET.178.236:494 SYN ******S*
Sep  2 06:24:36 208.26.55.145:42211 -> MY.NET.178.236:628 SYN ******S*
Sep  2 06:24:36 208.26.55.145:42211 -> MY.NET.178.236:482 SYN ******S*
Sep  2 06:24:36 208.26.55.145:42210 -> MY.NET.178.236:722 SYN ******S*
Sep  2 06:24:36 208.26.55.145:0 -> MY.NET.178.236:0 UNKNOWN 1**A**S*
RESERVEDBITS
Sep  2 06:24:36 208.26.55.145:42211 -> MY.NET.178.236:679 SYN ******S*
Sep  2 06:24:36 208.26.55.145:42211 -> MY.NET.178.236:407 SYN ******S*
Sep  2 06:24:36 208.26.55.145:40096 -> MY.NET.178.236:2048 NOACK *****RS*
Sep  2 06:24:36 208.26.55.145:42210 -> MY.NET.178.236:919 SYN ******S*
```

[...]

alerts :

[...]
09/02-06:24:10.614867 [**] Tiny Fragments - Possible Hostile Activity [**]
208.26.55.145 -> MY.NET.178.236
09/02-06:24:10.634319 [**] Tiny Fragments - Possible Hostile Activity [**]
208.26.55.145 -> MY.NET.178.236
09/02-06:24:10.634453 [**] Tiny Fragments - Possible Hostile Activity [**]
208.26.55.145 -> MY.NET.178.236
09/02-06:24:10.675351 [**] Tiny Fragments - Possible Hostile Activity [**]
208.26.55.145 -> MY.NET.178.236
09/02-06:24:10.675486 [**] Tiny Fragments - Possible Hostile Activity [**]
208.26.55.145 -> MY.NET.178.236
09/02-06:24:10.695164 [**] Null scan! [**] 208.26.55.145:59979 ->
MY.NET.178.236:0
09/02-06:24:10.695232 [**] Null scan! [**] 208.26.55.145:59979 ->
MY.NET.178.236:0
09/02-06:24:10.715017 [**] Null scan! [**] 208.26.55.145:59979 ->
MY.NET.178.236:0
09/02-06:24:10.715084 [**] Null scan! [**] 208.26.55.145:59979 ->
MY.NET.178.236:0
09/02-06:24:10.735321 [**] Null scan! [**] 208.26.55.145:59979 ->
MY.NET.178.236:0
[...]

OOS flag combination :

Sep 2 06:24:15 208.26.55.145:0 -> MY.NET.178.236:0 NOACK *2U***S*
RESERVEDBITS
Sep 2 06:24:36 208.26.55.145:0 -> MY.NET.178.236:0 UNKNOWN 1**A**S*
RESERVEDBITS
Sep 2 06:24:36 208.26.55.145:40096 -> MY.NET.178.236:2048 NOACK *****RS*
Sep 2 06:24:59 208.26.55.145:42214 -> MY.NET.178.236:109 NOACK **U*P*S*
Sep 2 06:25:01 208.26.55.145:42214 -> MY.NET.178.236:323 NOACK **U*P*S*

The source IP may have been spoofed. However, during the 5 days analysis period, only 208.26.55.145 has targeted MY.NET.178.236. And generating this kind of traffic would not give any information to the attacker if the source address was spoofed.

information on 208.26.55.145 :

```
reverse DNS : mail.geray.com

Trying whois -h whois.arin.net 208.26.55.145
Sprint (NETBLK-SPRINTLINK-BLKS) SPRINTLINK-BLKS 208.0.0.0 -
208.35.255.255
GE-RAY FABRICS, INC (NETBLK-FON-349137908879051) FON-349137908879051
208.26.55.144 - 208.26.55.151


whois -h whois.arin.net NETBLK-FON-349137908879051
GE-RAY FABRICS, INC (NETBLK-FON-349137908879051)
705 GINESI DR
MORGANVILLE, NJ 07751
US

Netname: FON-349137908879051
Netblock: 208.26.55.144 - 208.26.55.151

Coordinator:
KENNEY, GRANT (GK324-ARIN) gkenny@MONMOUTH.COM
(732)972-4033

Record last updated on 24-May-2001.
Database last updated on 9-Oct-2001 23:15:51 EDT.
```

SCANS

The scan log is recording UDP traffic from network games (half life or MSN gaming), creating false positive as a matter of scans. The gaming traffic between a limited amount of hosts has been removed to compute the top 10. we removed 64.37.156.9 talking UDP with 43 different machines on MY.NET, mainly ports 4960-4965 (5564 logs) I do not know what kind of application is using this range of UDP ports. Then we removed 216.162.3.20:7777 talking UDP to 4 different hosts (14872 logs) 6970/UDP has been removed. It looks like there could be about 30 realplayer servers on the network(6588 logs). A deeper investigation should be conducted as this could also be trojan activity.

top 10 scans

Number of scans	Port	Comments
30382	21/tcp	ftp. Several vulnerabilities have been discovered in ftp servers (WU-ftp...). Malicious users may also look for ftp which allows anonymous access to store files.
6989	80/tcp	http. part of the IIS worms. Several vulnerabilities. Some search engines are also scanning port 80 in order to find new web servers.
2461	3128/	squid-http. Squid is a proxy-cache that can be used as an anonymous

	tcp	proxy if not properly configured.
2059	1/rcp	tcpmux
1723	0	Indication of crafted packet. No legitimate traffic should use this port.
854	5/tcp	rje : remote job entry
271	111/ tcp	portmapper : directory for rpc services. Some rpc services have known vulnerabilities. 111/tcp should be blocked at the border.
224	1214/ tcp	kazaa file sharing, according to Andreas Östling in post http://www.incidents.org/archives/intrusions/msg00530.html
164	6346/ tcp	gnutella default port
124	515/ tcp	printer multi vendor vulnerability has been discovered on line printer daemon. http://xforce.iss.net/alerts/advise94.php

top 10 external source for scans :

Rank	Number of attempts	Source address
1	15469	212.199.28.76
2	6446	217.128.232.163
3	5949	205.188.246.121
4	5226	210.95.106.2
5	4711	130.89.229.75
6	2458	130.161.37.101
7	2200	217.11.167.47
8	2066	203.248.38.27
9	2050	130.239.36.121
10	1315	208.26.55.145

SCAN ANALYSIS #1

212.199.28.76 SYN scanned 15469 different hosts for 21/tcp

Sep 2 10:04:16 212.199.28.76:3061 -> MY.NET.244.6:21 SYN ******S*
Sep 2 10:04:16 212.199.28.76:3057 -> MY.NET.244.3:21 SYN ******S*
Sep 2 10:04:16 212.199.28.76:3056 -> MY.NET.244.2:21 SYN ******S*
Sep 2 10:04:16 212.199.28.76:3181 -> MY.NET.244.126:21 SYN ******S*

```
Sep  2 10:04:16 212.199.28.76:3175 -> MY.NET.244.120:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3174 -> MY.NET.244.119:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3170 -> MY.NET.244.115:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3171 -> MY.NET.244.116:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3166 -> MY.NET.244.111:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3163 -> MY.NET.244.108:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3161 -> MY.NET.244.106:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3149 -> MY.NET.244.94:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3141 -> MY.NET.244.86:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3122 -> MY.NET.244.67:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3121 -> MY.NET.244.66:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3111 -> MY.NET.244.56:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3107 -> MY.NET.244.52:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3094 -> MY.NET.244.39:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3093 -> MY.NET.244.38:21 SYN ******S*
Sep  2 10:04:16 212.199.28.76:3091 -> MY.NET.244.36:21 SYN ******S*
```

Registration information :

```
212.199.28.76 has no reverse DNS configured.


Trying whois -h whois.ripe.net 212.199.28.76
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum: 212.199.28.0 - 212.199.28.255
netname: GOLDENLINES
descr: DAILUP-PT
country: IL
admin-c: DR5299-RIPE
tech-c: DR5299-RIPE
status: ASSIGNED PA
notify: lir@linux.goldenlines.net.il
mnt-by: RIPE-NCC-NONE-MNT
changed: lir@linux.goldenlines.net.il 20010226
changed: lir@linux.goldenlines.net.il 20010626
source: RIPE

route: 212.199.0.0/16
descr: Golden Lines
origin: AS9116
mnt-by: AS9116-MNT
changed: lir@linux.goldenlines.net.il 20010807
source: RIPE

role: DNS REG
address: 25 Hsivim st. Petach-Tiikva, Israel
```

```
e-mail: dnsreg@012.net.il
trouble: dnsreg@012.net.il
admin-c: OM2369-RIPE
tech-c: GE2074-RIPE
nic-hdl: DR5299-RIPE
notify: lir@linux.goldenlines.net.il
changed: lir@linux.goldenlines.net.il 20001126
source: RIPE
```

Correlation with alert file : 48 Anonymous ftp attempts at the same time from the same source. It looks like the attacker is running a program that tries anonymous access to detected ftp servers. An anonymous access is interesting for an attacker to store files or to launch attacks with the "PORT" command.

SCAN ANALYSIS #2

130.89.229.75 scanned 4711 different hosts on MY.NET and attacked 203 machines during the same period of time

scan on port 80/tcp (web) with attacks on web sites found. The pattern of destination addresses shows this is not code red. Some target addresses are missing in the range. It may be a mechanism to evade from naive port scan detectors. It may also be an indication that some previous reconnaissance work has been conducted.

```
[...]
Sep  2 14:01:54 130.89.229.75:4462 -> MY.NET.177.162:80 SYN ******S*
Sep  2 14:01:54 130.89.229.75:4463 -> MY.NET.177.163:80 SYN ******S*
Sep  2 14:01:54 130.89.229.75:4476 -> MY.NET.177.175:80 SYN ******S*
Sep  2 14:01:54 130.89.229.75:4479 -> MY.NET.177.177:80 SYN ******S*
Sep  2 14:01:54 130.89.229.75:4481 -> MY.NET.177.179:80 SYN ******S*
Sep  2 14:01:54 130.89.229.75:4483 -> MY.NET.177.181:80 SYN ******S*
[...]
```

```
09/02-14:08:43.010757  [**] spp_portscan: PORTSCAN DETECTED from 130.89.229.75 (THRESHOLD 4 connections exceeded in 0 seconds) [**]
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:46.990663  [**] WEB-MISC Attempt to execute cmd [**] 130.89.229.75:4151 -> MY.NET.2.26:80
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
```

```
09/02-13:59:47.004454  [**] spp_http_decode: IIS Unicode attack detected [**] 130.89.229.75:4158 -> MY.NET.2.33:80
09/02-13:59:47.004454  [**] WEB-MISC Attempt to execute cmd [**] 130.89.229.75:4158 -> MY.NET.2.33:80
[...]
```

This detect can only occur after the TCP 3 way handshake occured. Very low probability of spoofed source IP.

registration information :

```
130.89.229.75 has valid reverse DNS of cal034031.student.utwente.nl

Trying whois -h whois.arin.net 130.89.229.75
University Twente (NET-UTNET)
Postbox 217
7500 AE Enschede
NL

Netname: UTNET
Netblock: 130.89.0.0 - 130.89.255.255

Coordinator:
Meijerink, Gert A. (GAM32-ARIN) G.A.Meijerink@CIV.UTWENTE.NL
+31 53 489 2326

Domain System inverse mapping provided by:

DINKEL.CIV.UTWENTE.NL 130.89.1.2
DRIENE.STUDENT.UTWENTE.NL 130.89.220.2
NS1.SURFNET.NL 192.87.106.101

Record last updated on 27-Mar-1996.
Database last updated on 10-Oct-2001 23:25:01 EDT.
```

SCAN ANALYSIS #3

130.161.37.101

```
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.164:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.183:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.186:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.192:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.215:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.227:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.233:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.250:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.251:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.252:3128 SYN **S*****
Sep  4 06:02:49 130.161.37.101:65535 -> MY.NET.1.254:3128 SYN **S*****
```

```
Sep  4 06:02:51 130.161.37.101:65535 -> MY.NET.2.131:3128 SYN **S*****
Sep  4 06:02:51 130.161.37.101:65535 -> MY.NET.2.147:3128 SYN **S*****
Sep  4 06:02:51 130.161.37.101:65535 -> MY.NET.2.149:3128 SYN **S*****
```

The source port set to 65535 (all bits set to 1) does not change. This proves that we are receiving crafted packets. 3128/tcp is squid proxy cache port. The attacker is trying to find misconfigured proxies in order to hide his identity for further attacks or web browsing.

Registration information

```
130.161.37.101 has valid reverse DNS of ntcarme.its.tudelft.nl

Trying whois -h whois.arin.net 130.161.37.101
Technische Universiteit Delft (NET-DUT-LAN)
Dienst Technische Ondersteuning
2600 AJ Delft,
NL

Netname: DUNET
Netblock: 130.161.0.0 - 130.161.255.255

Coordinator:
Kruijf, Freek de (FD18-ARIN) SSC@TUDelft.nl
+31 15 2783226 (FAX) +31 15 2783787

Domain System inverse mapping provided by:

NS1.TUDELFT.NL 130.161.180.1
NS2.TUDELFT.NL 130.161.180.65
NS1.SURFNET.NL 192.87.106.101
NS1.ET.TUDELFT.NL 130.161.33.17

Record last updated on 10-Nov-2000.
Database last updated on 11-Oct-2001 23:21:45 EDT.

The ARIN Registration Services Host contains ONLY Internet
Network Information: Networks, ASN's, and related POC's.
Please use the whois server at rs.internic.net for DOMAIN related
Information and whois.nic.mil for NIPRNET Information.
```

Out Of Spec :

In addition to the scanning (null scan, Xmas...) and TCP OS fingerprinting (Nmap, Queso) already reported in the alert or scan files, the OOS file has two interesting records. Don’t Fragment DF and More Fragment MF bits are both set. This is obviously out of spec. Unfortunately, we do not have enough information to classify this activity. During the analysis period, no hostile activity has been detected from the two sources.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/03-08:32:16.701789 212.113.188.69 -> MY.NET.68.222
TCP TTL:111 TOS:0x0 ID:7549  DF MF
Frag Offset: 0x0   Frag Size: 0x1A
0D 33 1E 61 00 20 BA C1 90 06 02 10 04 B3 55 18  .3.a. ........U.
08 45 6E FE 73 2C AC 8C C1 FA                    .En.s,....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/04-13:26:27.238843 63.192.100.222 -> MY.NET.208.174
TCP TTL:110 TOS:0x0 ID:55902  DF MF
Frag Offset: 0x0   Frag Size: 0x22
29 E4 D6 8F AC D3 24 A6 F4 65 17 34 95 17 B8 9F  ).....$..e.4....
50 A0 31 41 97 B1 23 DA 78 15 64 FB 2E 65 6B 85  P.1A..#.x.d..ek.
AD 09
```

Correlation : Russel Fulton detected traffic with DF and MF set.
http://archives.neohapsis.com/archives/incidents/2001-07/0071.html

"*For some time now snort has been logging 'Tiny Fragments' coming from several different addresses. Here is a sample:*
Packet 1
TIME: 10:04:55.405457
LINK: 00:00:0C:46:5C:D1 -> 00:E0:1E:8E:31:71 type=IP
IP: 62.32.156.41 -> 130.216.112.2 hlen=20 TOS=48 dgramlen=20 id=5D09
MF/DF=1/1 frag=0 TTL=98 proto=TCP cksum=CE6E
TCP: port 0 -> 0 seq=0000000000 ack=0000000000
hlen=0 (data=0) UAPRSF=000000 wnd=28 cksum=50A9 urg=59666
DATA: <No data>

Packet 2
TIME: 10:04:55.481006 (0.075549)
LINK: 00:00:0C:46:5C:D1 -> 00:E0:1E:8E:31:71 type=IP
IP: 62.32.156.41 -> 130.216.112.2 hlen=20 TOS=48 dgramlen=20 id=5D12
MF/DF=1/1 frag=0 TTL=98 proto=TCP cksum=CE65
TCP: port 0 -> 0 seq=0000000000 ack=0000000000
hlen=0 (data=0) UAPRSF=000000 wnd=28 cksum=0F59 urg=30577
DATA: <No data>
Note More Fragments and Don't fragment are both set to 1??
The packets arrive in pairs, both to the same destination address.
Some sources send packets to just one destination others send them to many.
When I look in the argus logs I see a single RST packet and argus does not report that it was fragmented.
Any idea what is going on? "

We do not have the exact same kind of traffic. Our TOS value is 0

The detects may have been produced by a packet corrupted by the last router before the sensor. Unfortunately, we do not know if the sensor recomputes the checksum and we do not have enough information to do it manually. If this is not corruption, then it can be :

- a kind of reconnaissance technique.
- an attempt to crash the remote system

Malicious detects that require further analysis :

EXPLOIT x86 family.

96 internal hosts targeted. If the following hosts are Intel based, an investigation for break ins or suspicious activity should be done.

MY.NET.130.86	*MY.NET.234.50*	*MY.NET.217.54*	*MY.NET.201.202*
MY.NET.217.158	MY.NET.237.74	MY.NET.111.130	MY.NET.204.146
MY.NET.222.138	MY.NET.218.254	MY.NET.210.230	MY.NET.209.194
MY.NET.208.14	MY.NET.202.62	MY.NET.227.182	MY.NET.222.246
MY.NET.210.246	MY.NET.207.190	MY.NET.203.22	MY.NET.203.134
MY.NET.1.6	MY.NET.224.78	MY.NET.221.154	MY.NET.221.106
MY.NET.221.10	MY.NET.212.114	MY.NET.210.134	MY.NET.209.246
MY.NET.209.174	MY.NET.206.74	MY.NET.205.94	MY.NET.205.10
MY.NET.203.238	MY.NET.181.76	MY.NET.97.198	MY.NET.237.94
MY.NET.236.222	MY.NET.235.54	MY.NET.235.14	MY.NET.234.46
MY.NET.233.202	MY.NET.233.146	MY.NET.229.66	MY.NET.228.182
MY.NET.226.214	MY.NET.225.34	MY.NET.225.114	MY.NET.223.22
MY.NET.223.18	MY.NET.222.106	MY.NET.221.94	MY.NET.221.22
MY.NET.221.210	MY.NET.221.150	MY.NET.220.90	MY.NET.220.122
MY.NET.219.214	MY.NET.219.130	MY.NET.218.174	MY.NET.218.158
MY.NET.217.66	MY.NET.217.46	MY.NET.217.106	MY.NET.217.102
MY.NET.212.46	MY.NET.212.230	MY.NET.211.114	MY.NET.209.90
MY.NET.209.70	MY.NET.209.162	MY.NET.208.90	MY.NET.208.146
MY.NET.207.34	MY.NET.207.182	MY.NET.206.246	MY.NET.206.114
MY.NET.206.110	MY.NET.205.98	MY.NET.205.78	MY.NET.205.254
MY.NET.205.242	MY.NET.204.118	MY.NET.204.10	MY.NET.203.70
MY.NET.203.26	MY.NET.203.138	MY.NET.203.114	MY.NET.202.86
MY.NET.201.78	MY.NET.201.54	MY.NET.201.226	MY.NET.20.10
MY.NET.178.115	MY.NET.163.97	MY.NET.153.162	MY.NET.153.160

RPC tcp traffic contains bin_sh :

146.186.15.46 -> MY.NET.229.178

Back Door netmetro :

Two machines should be checked quickly : MY.NET.179.77 and MY.NET.5.29
reference: http://www.whitehats.com/IDS/79

Virus, possible worm :

MY.NET.6.44 is a pop-3 server. Infected mail may reside on this server.

possible warez site :

The following ftp servers should be examined :MY.NET.111.159 MY.NET.138.205, MY.NET.139.169, MY.NET.144.59, MY.NET.144.38, MY.NET.151.88, MY.NET.178.108 and MY.NET.162.205 . Look for hidden directory, use `ls -al` to list files. The university may want to avoid copyright violation and inappropriate use of its network resources.

General recommendations :

Install a firewall if you do not have one. Start to block unused or dangerous ports (NetBios, sunrpc, echo, chargen...). Restrict incoming DNS traffic to your DNS servers. Same thing for smtp and ftp. Look at your firewall logs. List the services that are used and if you can restrict them. The more services you can restrict, the better. One day, you will be able to have a set of firewall rules that ends with "any -> any, service:any : drop".

Maintain up to date anti-viruses software on servers AND workstations.

Ask your management if gaming is considered appropriate use of the network.

Analysis process

I first tried to use snortsnarf http://www.silicondefense.com/software/snortsnarf/index.htm, a snort log processor. Unfortunately, my Linux Laptop ran out of memory. On smaller log files, I did not get what I expected. Instead of spending too much time trying to understand snortsnarf, I manipulated directly the logs. My tools of choice are : grep, awk, sed, sort, cat, wc, head, tail and uniq. Most of the work has been done with series of these simple and efficient tools. It may not be the most efficient way of analysing logs, but you have to know exactly what you are doing and you also know exactly what the data you extracted comes from and what they mean.

Alert file format :

date [**] description of attack [**] source[:port] -> dest[:port]

Scan file format :

date (3 fields) source:port -> dest:port additional information (UDP or TCP flags)

I did not write script files. Command line editing in bash is powerful enough to recall a full “for” loop and edit it. When a serie of commands was proved useful, I just copy and paste it in a text file for future use.

Concatenation of 5 alert files into one :

```
cat alert.* > alert
```

remove start and stop comments as well as portscan alerts.

```
grep '\[\*\*\]' alert | grep -v spp_portscan > al
```

How to create top 10 attacks. The sed command transforms [**] into **, does someone know how to tell awk that separator field is “[**]” ?

```
cat al | sed -e 's/\[\*\*\]/\*\*/g' |awk -F'**' '{print $2}' | sort |
uniq -c | sort -nr > attack.sorted
```

top 10 source IP and destination IP :

```
cat al |awk -F'**' '{print $3}' | awk '{print $2}' | awk -F':' '{print
$1}' |sort | uniq -c | sort -nr > srcips_sorted

cat al |awk -F'**' '{print $3}' | awk '{print $4}' | awk -F':' '{print
$1}' |sort | uniq -c | sort -nr > dstips_sorted
```

Trimming the scan logs

```
 grep -e'^Sep' scans |awk '{print $4" "$6" "$7" "$8}' > allscans
```

sorting top source scans

```
awk '{print $1}' allscans | awk -F':' '{print $1}' |sort | uniq -c |
sort -nr |head -10> src_scans
```

sorting top 10 destination port scans

```
awk '{print $2}' allscans | awk -F':' '{print $1}' |sort | uniq -c |
sort –nr|head -10 > dst_scans
```

sorting top dest port in scans :

```
awk '{print $2}' allscans | awk -F':' '{print $2}' |sort | uniq -c |
sort -nr > dstport_scans
```

List of alerts for which an IP is involved. src_i contains the list of IPs to check

```
for i in `cat src_i`; do echo $i >>list.txt ; grep $i alerts | sed -e
's/\[\*\*\]/\*\*/g' |awk -F'**' '{print $2}' | sort | uniq -c | sort -nr
>> list.txt; done
```

References:

http://www.samspade.org is a useful site to get information on IP addresses. DNS requests do not come from your site.

http://www.google.com is a very efficient search engine. Powerful for correlation.

http://www.incidents.org gives valuable information on ongoing security threats and technical information, as well as discussions.

The SANS Institute. Network Traffic Analysis Using TCP Dump. Course reference Parliament Hill Aug 2001.

The SANS Institute TCP/IP for Firewalls and Intrusion Detection. Course reference Parliament Hill Aug 2001

The SANS Institute . IDS Signatures and Analysis. Course reference Parliament Hill Aug. 2001.

Last Updated: April 23rd, 2018

Upcoming Training

SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201805,	May 02, 2018 - Jun 14, 2018	vLive
Community SANS Virginia Beach SEC503	Virginia Beach, VA	May 07, 2018 - May 12, 2018	Community SANS
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
SANS Oslo June 2018	Oslo, Norway	Jun 18, 2018 - Jun 23, 2018	Live Event
Mentor Session - SEC503	Houston, TX	Jun 18, 2018 - Jul 18, 2018	Mentor
SANS Minneapolis 2018	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	Live Event
Minneapolis 2018 - SEC503: Intrusion Detection In-Depth	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	vLive
SANS London July 2018	London, United Kingdom	Jul 02, 2018 - Jul 07, 2018	Live Event
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
Security Operations Summit & Training 2018	New Orleans, LA	Jul 30, 2018 - Aug 06, 2018	Live Event
San Antonio 2018 - SEC503: Intrusion Detection In-Depth	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	vLive
SANS San Antonio 2018	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	Live Event
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS Virginia Beach 2018	Virginia Beach, VA	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS Amsterdam September 2018	Amsterdam, Netherlands	Sep 03, 2018 - Sep 08, 2018	Live Event
SANS Tokyo Autumn 2018	Tokyo, Japan	Sep 03, 2018 - Sep 15, 2018	Live Event
SANS London September 2018	London, United Kingdom	Sep 17, 2018 - Sep 22, 2018	Live Event
SANS Network Security 2018	Las Vegas, NV	Sep 23, 2018 - Sep 30, 2018	Live Event
SANS Brussels October 2018	Brussels, Belgium	Oct 08, 2018 - Oct 13, 2018	Live Event
SANS Northern VA Fall- Tysons 2018	Tysons, VA	Oct 13, 2018 - Oct 20, 2018	Live Event
SANS Denver 2018	Denver, CO	Oct 15, 2018 - Oct 20, 2018	Live Event
SANS October Singapore 2018	Singapore, Singapore	Oct 15, 2018 - Oct 28, 2018	Live Event
Mentor Session - SEC503	Ballston, VA	Nov 01, 2018 - Dec 06, 2018	Mentor
SANS Dallas Fall 2018	Dallas, TX	Nov 05, 2018 - Nov 10, 2018	Live Event
SANS San Diego Fall 2018	San Diego, CA	Nov 12, 2018 - Nov 17, 2018	Live Event
SANS Stockholm 2018	Stockholm, Sweden	Nov 26, 2018 - Dec 01, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced