Sans Institute

GIAC Intrusion Detection Curriculum

Monterey 2000 - Practical Assignment

Claudio R. G. Silotto

Assignment 1 – Network Detects

In this part of the assignment I'm presenting four network detects with comments and analysis. They were taken from different places, some from the wild, some from a test lab but none of them were hand made or forged.

Before I writing my analysis about these logs I spend some time trying to figure out what each line could mean, and then, I wrote my final decision. So you are always going to see a clear definition of which kind of attack created that trace, even though my decision may not be the best one!

Detect 1

```
Nov 12 07:28:45: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.227.84.248(59466) -> OUR.NET.248.241(53), 1 packet
Nov 12 07:29:38: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.227.84.248(59466) -> OUR.NET.248.242(53), 1 packet
Nov 12 07:29:44: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.227.84.248(59466) -> OUR.NET.248.241(53), 1 packet
Nov 12 07:30:26: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.227.84.248(59466) -> OUR.NET.248.242(53), 1 packet
Nov 12 09:16:55: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.176.67.124(60456) -> OUR.NET.248.35(53), 1 packet
Nov 12 09:17:13: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.176.67.124(60456) -> OUR.NET.248.35(53), 1 packet
Nov 12 14:27:02: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.227.95.10(43975) -> OUR.NET.248.1(53), 1 packet
(… lines cut… scan from .1 to .254…)
Nov 12 14:38:19: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.227.95.10(43975) -> OUR.NET.248.254(53), 1 packet
Nov 12 20:46:54: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.176.67.180(39391) -> OUR.NET.248.241(53), 1 packet
Nov 12 20:47:34: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.176.67.180(39392) -> OUR.NET.248.242(53), 1 packet
```

1. Source of trace:
I was able to get this trace from the production network of a small Web-Hosting company.

2. Detect was generated by:
This log came from a Cisco 7500 border router, which blocked the packets with its access-list (Inter-IN). Those are pretty common entries on logs, but I choose to use it on this paper due to the high increase of this kind of attack in the past weeks. The destination IP addresses have been changed to hide the original source.

The format of the log is the following:

```
Date    Time      Type of Alarm               list name    action  protocol  SourceIP  (Port)  flow  DestinationIP    (Port), Number of packets received
Nov 12 07:28:45: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.227.84.248(59466) -> OUR.NET.248.241(53), 1 packet
```

3. Probability the source address was spoofed

The probability that the source IP addresses of those packets are spoofed is low, since it's an attempt to do a TCP connection. To craft sequence numbers requires a hard work and that technique would be needed if the server starts responding. It is possible that the attacker was simply sending packets with somebody else IP address and didn't care about the three-way handshake but that doesn't make sense in this kind of attack unless the attacker was in the middle of the way and could watch the traffic. I really don't believe so and for me, they are not spoofed!

4. Description of attack:

These packets could means many things since there is a great number of attacks against DNS servers. We can see that there is a network scan in the middle of the log, but the other packets, the interesting ones, could also be the normal behavior of a DNS client asking for some information that exceeded the maximum size of a UDP DNS packet (484 bytes of data), but we don't have such big entries in the zone map of this site. So I bet my coins that this is the beginning of a Deny of Service attack, mainly because fabio@telemail.it announced on Nov 07th 2000 at Bugtraq a possible Deny of Service that works on most BIND 8.2.2-P5 implementations. And we know people love to try new things!

5. Attack mechanism:

If I'm right about those packets, the attack works as following: the attacker starts a TCP connection to port 53 on the DNS server, and asks for a compressed zone transfer. Receiving that, the server is suppose to crash after an authoritative and non-cached name server query is made. The only restriction to this attack is that the attacker must be an authorized host to do zone transfers in that server, but we still see many misconfigured DNS server out there so this attack still dangerous.

6. Correlations:

Internet Software Consortium – Bind Vulnerabilities page: http://www.isc.org/products/BIND/bind-security.html
CERT Advisory CA-2000-20 - http://www.cert.org/advisories/CA-2000-20.html
Security Focus – Bugtraq ID 1923 - http://www.securityfocus.com/bid/1923
Common Vulnerabilities and Exposures – Database version – 20001013 – No entries or candidates found. http://cve.mitre.org
John Springer GCIA Practical - DNS Host Scan - http://www.sans.org/y2k/practical/John_Springer.doc (more links can be found in there)

7. Evidence of active targeting:

Even though we can see that some of the packets are not going to our DNS (like in the scans), we have an active targeting, Yes!

8. Severity:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 5 (DNS server are critical to this kind of business)
Lethality = 5 (Deny of Service not only of the attacked machine, but also would leave other system unavailable)
System Countermeasures = 3 (Zone transfers are already restricted to many machines but we have some hosts that we have to allow; no patches for this bug is applied)
Network Countermeasures = 5 (Access-lists are tightly applied)

S = (5 + 5) – (3 + 5)
S = 2

9. Defensive recommendation:

The mechanisms to avoid this attack that I think should be implemented on this site are:

- Access-list on the border router to avoid any connection to the port 53 TCP - DONE!
- Restricted zone transfers on the DNS server as tight as possible (named.conf file) – DONE!
- Update to the latest version of BIND – Pending!

Those are good defenses, but also, would be nice to have some IDS sensor looking for UDP 53 traffic hitting the DNS servers, and warn on Version.Bind packets and other types of suspicious queries. Also a sensor looking for those TCP 53 packets would be very useful. I know it is difficult to place this sensor because we are filtering at the border router, but without it, it is hard to know what those packets are for. We need to know if some packet is a stimulus or not, if the attacker is launching the attack or is doing a hosts/port scan before we drop them, and for that, we need to see the IP and TCP header, flags and maybe take a look into the data portion of the packet. An option would be to allow those packets to pass the border router, and block then in a second security wall, probably a firewall. That way, we could place our sensor in this DMZ and have a better analysis.

10. Multiple choice test question:

If you see the next line in your Cisco Logging Buffer, you can say:

```
Nov 12 09:16:55: %SEC-6-IPACCESSLOGP: list Inter-IN denied tcp 200.176.67.124(60456) -> OUR.NET.248.35(53), 1 packet
```

a. This is an attack to our DNS server
b. This is a normal query from a DNS client
c. This is a scan to see if the DNS allow zone transfers
d. All lines above can be true

Correct Answer → letter "d"

Detect 2

```
Oct 20 10:19:52.737 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqqT sent=78 rcvd=185 srcif=lan1 src=200.226.36.162/1597 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/ result="200 OK" proto=http rule=2
Oct 20 10:19:52.758 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqqV sent=87 rcvd=1078 srcif=lan1 src=200.226.36.162/1598 dstif=lan5 dst=172.17.1.11/80 op=GET
arg=http://www.invalid.com.br/cgi-local/ result="404 Not found" proto=http rule=2
Oct 20 10:19:52.775 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqqX sent=81 rcvd=1078 srcif=lan1 src=200.226.36.162/1599 dstif=lan5 dst=172.17.1.11/80 op=GET
arg=http://www.invalid.com.br/cgi/ result="404 Not found" proto=http rule=2
Oct 20 10:19:52.794 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=Dqr0 sent=84 rcvd=1078 srcif=lan1 src=200.226.36.162/1600 dstif=lan5 dst=172.17.1.11/80 op=GET
arg=http://www.invalid.com.br/cgibin/ result="404 Not found" proto=http rule=2
Oct 20 10:19:52.810 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=Dqr2 sent=83 rcvd=1078 srcif=lan1 src=200.226.36.162/1601 dstif=lan5 dst=172.17.1.11/80 op=GET
arg=http://www.invalid.com.br/htbin/ result="404 Not found" proto=http rule=2
Oct 20 10:19:52.827 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=Dqr4 sent=82 rcvd=1078 srcif=lan1 src=200.226.36.162/1602 dstif=lan5 dst=172.17.1.11/80 op=GET
arg=http://www.invalid.com.br/cgis/ result="404 Not found" proto=http rule=2
(…cut lines to better fit…)
Oct 20 10:20:02.439 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=Dqyw sent=98 rcvd=125 srcif=lan1 src=200.226.36.162/1832 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/iisadmpwd/aexp2b.htr result="404 Not found" proto=http rule=2
Oct 20 10:20:02.453 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=Dqyy sent=97 rcvd=125 srcif=lan1 src=200.226.36.162/1833 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/iisadmpwd/aexp3.htr result="404 Not found" proto=http rule=2
Oct 20 10:20:02.467 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqyA sent=97 rcvd=125 srcif=lan1 src=200.226.36.162/1834 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/iisadmpwd/aexp4.htr result="404 Not found" proto=http rule=2
Oct 20 10:20:02.484 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqyC sent=98 rcvd=125 srcif=lan1 src=200.226.36.162/1835 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/iisadmpwd/aexp4b.htr result="404 Not found" proto=http rule=2
Oct 20 10:20:02.498 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqyE sent=96 rcvd=125 srcif=lan1 src=200.226.36.162/1836 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/iisadmpwd/anot.htr result="404 Not found" proto=http rule=2
Oct 20 10:20:02.512 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqyG sent=97 rcvd=125 srcif=lan1 src=200.226.36.162/1837 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/iisadmpwd/anot3.htr result="404 Not found" proto=http rule=2
```

1. **Source of trace:**

This trace was captured from a production firewall of a small Web-Hosting company. The same one from the Detect 1.

2. **Detect was generated by:**

This log came from a Praesidium e-Firewall 6.0.3, a HP Product that is similar to Axent Raptor.

The format of the log is the following:

```
Jun 20 10:19:52.737 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqqT sent=78 rcvd=185 srcif=lan1 src=200.226.36.162/1597 dstif=lan5 dst=172.17.1.11/80 op=HEAD
   A         B         C        D             E             F           G        H         I        J             K             L       M            N          O     P
arg=http://www.invalid.com.br/ result="200 OK" proto=http rule=2
             Q                      R               S       T
```

A – Date	H – Bytes Sent	O – Destination Port address
B – Timestamp	I – Bytes Received	P – Options
C – System name	J – Interface from where the packet came in	Q – Arguments from this connection
D – Daemon	K – IP address of the source	R – Result sent back
E – Event Type	L – Source Port address	S - Protocol
F – Duration of this connection	M – Interface from where the packet leaves	T – Rule matched
G – ID	N – Destination IP address	

Isn't this a cool log! Lots of info, I love it!

3. Probability the source address was spoofed

Unlikely. Those are all TCP connections and they needed to complete the three-way handshake. It is not easy to predict sequence numbers even more because our servers are running with the latest patches and have a pretty good randomized sequence numbers function.

4. Description of attack:

The attacker is looking for some common vulnerability on web servers. Basically, the client asks many files for the server and see which one is available, so he can exploit it later. In this case, the attacker already knew that we had a server running on that IP address, because we couldn't find any correlation about network or port scan from that IP, but his reconnaissance was not really good, and we can see some attacks that are not design to our type of Operational System or Web Server Application. Another possibility is that the attacker didn't know or wanted to change the script and simply send every type of vulnerability test.

5. Attack mechanism:

As we said before, this is a scripted attack. We can say that because we had more than 100 events over 1 minute all coming from the same source. There are many programs available nowadays that do those kinds of vulnerability scans, even commercial products, like Retina and ISS Internet Scanner. Those programs are frequently updated so its difficult to say exactly which program launched the attack. Two of the most commons ones are Whisker and VLAD. Those programs exploit misconfigurations on web-servers, bugs and bad coding of CGI scripts.

6. Correlations:

Cerberus' Internet Scanner - http://www.cerberus-infosec.co.uk/cis.shtml
VLAD - http://razor.bindview.com/tools/vlad/index.shtml
Whisker - http://www.wiretrip.net/rfp/p/doc.asp?id=21&iface=2
Retina – eEye: http://www.eeye.com/html/Products/Retina/overview.html
Lenny Zeltser GCIA Practical - http://www.sans.org/y2k/practical/Lenny_Zeltser.htm

7. Evidence of active targeting:

Sure we have… All the packets are hitting a single host and service.

8. Severity:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 4 (Web-Server is very important for this business)
Lethality = 4 (We don’t know which vulnerability the attacker can find – some of them can give root access so I used a average value)
System Countermeasures = 3 (The administrator of this site doesn’t care too much about pretty new patches)
Network Countermeasures = 1 (Router and firewall allow those packets to pass)

S = (4 + 4) – (3 –1)
S = 4

9. Defensive recommendation:

- Install up to date patches in our servers.
- Do some bastion configuration on the web servers and restrict permissions of files and directories.
- Do a security review of all CGI scripts (ASP, Java, whatever).

10. Multiple choice test question:

Looking at the following log, we can say:

```
Oct 20 10:20:02.512 efw01 httpd[9914]: 121 Statistics: duration=0.00 id=DqyG sent=97 rcvd=125 srcif=lan1 src=200.226.36.162/1837 dstif=lan5 dst=172.17.1.11/80 op=HEAD
arg=http://www.invalid.com.br/iisadmpwd/anot3.htr result="404 Not found" proto=http rule=2
```

a. We would be safe from this attack using a better access-list policy to block those invalid requests to our web servers
b. There is no attack known related anyhow to this log
c. This a NAT related attack
d. None of above

Correct Answer → letter “d”

Detect 3

```
11:46:12.048190 200.xxx.xxx.11.80 > xxx.xxx.xxx.254.15090: R 0:0(0) ack 439964754 win 0
11:46:17.484755 200.xxx.xxx.11.80 > xxx.xxx.xxx.253.32753: R 0:0(0) ack 440639696 win 0
11:46:24.653312 200.xxx.xxx.10.80 > xxx.xxx.xxx.188.21476: R 0:0(0) ack 2574828510 win 0
11:46:34.353759 200.xxx.xxx.10.80 > xxx.xxx.xxx.187.15320: R 0:0(0) ack 2575788208 win 0
11:46:49.030915 200.xxx.xxx.10.80 > xxx.xxx.xxx.186.34313: R 0:0(0) ack 2577705619 win 0
11:46:52.711900 200.xxx.xxx.10.80 > xxx.xxx.xxx.185.28899: R 0:0(0) ack 2578323993 win 0
11:46:59.813068 200.xxx.xxx.10.80 > xxx.xxx.xxx.184.22286: R 0:0(0) ack 2579048134 win 0
11:47:06.117277 200.xxx.xxx.10.80 > xxx.xxx.xxx.183.28222: R 0:0(0) ack 2579657523 win 0
11:47:12.015013 200.xxx.xxx.11.80 > xxx.xxx.xxx.252.23368: R 0:0(0) ack 447351829 win 0
11:47:16.034577 200.xxx.xxx.11.80 > xxx.xxx.xxx.251.34484: R 0:0(0) ack 447930756 win 0
```

1. **Source of trace:**

A Test Network, with an external router, some hosts in the DMZ, a firewall doing NAT and a few servers behind.

2. **Detect was generated by:**

Tcpdump running on a sensor placed at the DMZ of our test network.

The format of the log is the following:

```
11:47:16.034577 200.xxx.xxx.11.80 > xxx.xxx.xxx.251.34484: R 0:0(0) ack 447930756 win 0
    Time      Source_IP  Port Dest_IP     Port Flags Ack Number  WindowSize
```

3. **Probability the source address was spoofed:**

Low. There is a small chance that this attack works with spoofed IP address, yes it can be done like that but it only makes sense if you are in the middle of the way of the returning packets, so you can see the response from the router or server and keep building you scan table.

4. **Description of attack:**

This is an Inverse Network Mapping scan. The attacker can map our network doing an analysis of the responses from our network devices to those RESET packets. The point here is that those are Resets to connections that never existed but the router doesn't know that and will allow those packets to get into our network, thinking that would be valid responses. If the destination IP doesn't exists, then the router will respond with an ICMP host unreachable message (ICMP type 3-1).

5. **Attack mechanism:**

It's pretty neat the way this scan works. Let's start looking at a common scenario.

Normally we allow machines from the inside of our network to surf the web, and may times, someone asks for a page to a server that is not running a web server at that particular time. So, what is going to happen? We would see a SYN packet leaving our network from

client.inside.net/HIGHPORT and going to www.invalid.net/80. That server doesn't have port 80 open, so it will answer back with a RESET to that TCP connection. That's exactly our trace, of course without the SYN packet, it never occurred.
As we know, packet filters devices are not statefull, so they don't know if there was any solicitation to that connection before, and if there is no access-list blocking that packet, they will forward the packets.

6. Correlations:

Many RESET scans were first seen in the week of Sep 13th 1998. The interesting point about that scan is that they came from multiple ISPs and had identical ACK numbers. Nowadays, we can't trap an alarm based on that since they are changing ACK numbers. Even I in this paper handcraft some packets with different sequence numbers. So the only signature for this scan is RESET flag with no other activity such as a SYN before.

7. Evidence of active targeting:

We can say that was a target in this scan, it was our hosts in the DMZ. I know that network mapping is not the best example of a target attack but what I'm trying to say is that this was not a random scan or anything like that, the attacker was trying to get some confidential information about our site.

8. Severity:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 5 (We have some critical hosts with valid IP addresses in the DMZ, like DNS, web-servers and mail relay)
Lethality = 1 (We only loose confidentiality with this scan, and it's unlikely to succeed)
System Countermeasures = 1 (We don't have many things that can help us here, this is the way TCP/IP stack works)
Network Countermeasures = 5 (Router is dropping packets silently)

S = (5 + 1) – (1 + 5)
S = 0

9. Defensive recommendation:

First, disable ICMP responses from your routers. Doing that, you are pretty much safe from that kind of attack. Imagine, dropping the packets because the destination host doesn't exist and not sending any ICMP message back, is the same as forwarding it to the destination. So the resulting scan from that attacker would be useless for him because there is no way to identify which hosts exists or not.

A good paper on how to secure Cisco routers and disable those kinds of answer was written by brett (beldridg@best.com) and variable-k (variablek@home.com) and can be found at Phrack Magazine volume 55 http://www.phrack.com. I'm not saying that you have to go and implement access-lists in all your routers, I'm saying that you should have a secure configuration on it.

10. Multiple choice test question:

To do an inverse scan using TCP, which flags should be on in a packet?

a. SYN
b. RST
c. SYN/FIN
d. FIN

Correct Answer → letter "b"

Detect 4

```
 [**] Tiny Fragments - Possible Hostile Activity [**]
11/11-12:35:52.623053 200.xxx.xxx.10 -> xxx.xxx.xxx.99
ICMP TTL:238 TOS:0x0 ID:31337  MF
Frag Offset: 0x0   Frag Size: 0x2B
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] Tiny Fragments - Possible Hostile Activity [**]
11/11-12:35:52.623140 200.xxx.xxx.10 -> xxx.xxx.xxx.99
ICMP TTL:238 TOS:0x0 ID:31337  MF
Frag Offset: 0x0   Frag Size: 0x2B
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] Tiny Fragments - Possible Hostile Activity [**]
11/11-12:35:52.628419 200.xxx.xxx.10 -> xxx.xxx.xxx.99
ICMP TTL:238 TOS:0x0 ID:31337  MF
Frag Offset: 0x0   Frag Size: 0x2B
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] Tiny Fragments - Possible Hostile Activity [**]
11/11-12:35:52.628505 200.xxx.xxx.10 -> xxx.xxx.xxx.99
ICMP TTL:238 TOS:0x0 ID:31337  MF
Frag Offset: 0x0   Frag Size: 0x2B
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

1. **Source of trace:**

The same test network used on Detect 3.

2. **Detect was generated by:**

Snort 1.6.3 placed at our DMZ with a basic set of rules.

The format of the log is the following:

```
[**] Tiny Fragments - Possible Hostile Activity [**]
                          A
11/11-12:35:52.628505 200.xxx.xxx.10 -> xxx.xxx.xxx.99
  B          C              D                E
ICMP TTL:238 TOS:0x0 ID:31337  MF
  F     G       H       I      J
Frag Offset: 0x0   Frag Size: 0x2B
      K                  L
```

A – Alert Message	G – TTL Value
B – Date	H – Type of Service Value
C – Time	I – ID of the Fragment

D - Source IP Address	J – Flags
E – Destination IP Address	K – Fragment Offset Value
F – Protocol	L – Fragment Total Size

3. Probability the source address was spoofed
High! There is no meaning for an attacker put his IP address in this attack. As long as the routers in the middle can route correctly to the destination, the source IP address doesn't matter much.

4. Description of attack:
Fragment Alarms can mean many things. It can be a simple attack that exploits the TCP/IP stack of the destination host (like Ping of Death, Teardrop and SSping), or it can be a strategy from the attacker to deceive Intrusion Detection Systems (like Fragrouter). Doing this, an attacker can split the information inside the packet so they don't match (or do match!) a defined template. Someone could send a packet opening a connection to port 80 TCP and then, right after this packet been accepted by a filtering device, send another fragment and re-write the TCP header to port 23. Or, someone can make multiples fragments to send the string "/etc/passwd" with a single letter in each fragment. Many IDS will not complain about "/", "e", "t", "c", "/", "p", "a", "s", "s", "w", "d" even thought they are watching for the content "/etc/passwd". Just to give you an idea, Snort (www.snort.org) is going to do full fragmentation reassembly only in version 2.0. Now there is only a rule option to watch for Tiny Fragments and will alarm on any packet smaller than 256 bytes (what our alarm means).

5. Attack mechanism:
In this case, we can highlight the fact that the protocol is ICMP, and the offset is 0x0. So probably this is an ICMP type of packet that we are allowing to pass through our filtering devices. This is probably a ssping packet because the ID of the fragments doesn't change and is equal to 31337.

6. Correlations:
Ping of Death – CVE-1999-0128 – http://cve.mitre.org
Teardrop – CVE-1999-0015 – http://cve.mitre.org
Fragrouter – http://www.anzen.com/research/nidsbench/fragrouter.html
Tiny Fragments - Brian Betterton – GCIA - http://www.sans.org/y2k/practical/brian_betterton.doc

7. Evidence of active targeting:
Yes, we can see one clear target because all the packets are going to the same host.

8. Severity:
Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 5 (This attack can reach any of our hosts, including DNS, web servers and mail relay)
Lethality = 4 (It can result in a Deny of Service)

System Countermeasures = 5 (We are running new versions of OS and patches are applied)
Network Countermeasures = 1 (we can't do much, fragments can pass through packet filters)

S = (5 + 4) – (5 – 1)
S = 3

9. Defensive recommendation:

- Keep up to date with the necessary patch in all hosts.
- Always use IDS Sensors to alarm on small fragments.

10. Multiple choice test question:

A good way to identify malicious fragments and most of the Intrusion Detection Systems are using nowadays is:

a. Looking at the TOS specified
b. Looking at the Protocol
c. Looking at the Fragment Size
d. Looking at the TTL

Correct Answer → letter "b"

Assignment 2

Evaluate an Attack

1. Introduction

I choose to comment about a new Denial of Service attack against Microsoft Exchange Server 5.5 that recently came out. This is not a complex attack, far from that, but those are the kind of attacks that scary me, because anyone can quickly learn and go out playing with it.

Basically it works sending an unexpected value in one of the fields in the MIME header. When Microsoft Exchange receives that message and starts to process this email, Exchange crash. To solve this problem, an administrator has to log in that server, find the problematic mail, erase it and restart the service.

2. Description

To launch this attack all you need is a Telnet client program and the name (or IP address) of the target server. That's it!

Once you have that, you have to do a telnet to port 25 of the target server. Start sending all the basic fields of an email, like sender email address and receiver email address. Then, start sending the MIME header, version, content-type and then you need to pass the charset utilized in your email, so, send nothing! That's the trick! Finish your email, and you are done!

The next lines show an example of lines that you could possible do to send an attack like this:

```
# telnet mailserver.invalid.net 25
```

```
MAIL FROM: notyourname@someplace.net
RCPT TO: administrator
DATA

MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary=”=_ Boundary 1-KtwEv4jY84Hk”
--=_ Boundary 1-KtwEv4jY84Hk
Content-Type: text/plain;
charset = “”
Content-Transfer-Encoding: 7bit
```

```
This is your message
--=_ Boundary 1-KtwEv4jY84Hk –

<CRLF> . <CRLF>

quit
```

* note that <CRLF> means an empty line

3. How it looks in the network

If we start thinking how would this attack looks in the network, we realize that there is not a big difference from this email to any other valid email. And this is a problem. We cannot block a TCP connection to port 25 otherwise we couldn't receive any message. Ok, maybe there is a DNS query before the attack packets, but that is also very normal. No router can help us here. Maybe a proxy could. What we can do, is to watch all the packets hitting our Exchange Server, and Alarm when we see a pattern.

Here is an example of Snort Rule that worked in my test lab.

```
alert TCP any any -> $HOME_NET 25: (msg; “Microsoft Exchange 5.5 -  Malformed MIME Header”; content:  “charset=\”\””; nocase; )
```

where $HOME_NET is the subnet of my Exchange servers in the format x.y.w.z/C

Of course this rule will slow down Snort, since it is looking for packet's payload, but it is a way to trap an Alarm when those packets come.

Following is the output of Snort 1.6.3. I used the –O option to hide the IP address of the server, and cut some packets after the three-way handshake to save some space in this paper.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:52:57.291031 200.226.10.123:1034 -> xxx.xxx.xxx.xxx:25
TCP TTL:64 TOS:0x0 ID:142  DF
**S***** Seq: 0x340DC19A   Ack: 0x0   Win: 0x7F88
TCP Options => MSS: 1484 SackOK TS: 239138 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:52:57.404850 xxx.xxx.xxx.xxx:25 -> 200.226.10.123:1034
TCP TTL:120 TOS:0x0 ID:56936  DF
**S***A* Seq: 0x2912C650   Ack: 0x340DC19B   Win: 0x2238
```

```
TCP Options => MSS: 1460

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:52:57.405031 200.226.10.123:1034 -> xxx.xxx.xxx.xxx:25
TCP TTL:64 TOS:0x0 ID:143  DF
******A* Seq: 0x340DC19B   Ack: 0x2912C651   Win: 0x7F88

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:52:57.554844 xxx.xxx.xxx.xxx:25 -> 200.226.10.123:1034
TCP TTL:120 TOS:0x0 ID:57448  DF
*****PA* Seq: 0x2912C651   Ack: 0x340DC19B   Win: 0x2238
32 32 30 20 6D 61 69 6C 2D 73 70 2E XX XX XX XX  220 mail-sp.XXXX
XX XX XX 2E XX XX XX 2E 62 72 20 45 53 4D 54 50  XXX.XXX.br ESMTP
20 53 65 72 76 65 72 20 28 4D 69 63 72 6F 73 6F   Server (Microso
66 74 20 45 78 63 68 61 6E 67 65 20 49 6E 74 65  ft Exchange Inte
72 6E 65 74 20 4D 61 69 6C 20 53 65 72 76 69 63  rnet Mail Servic
65 20 35 2E 35 2E 32 34 34 38 2E 30 29 20 72 65  e 5.5.2448.0) re
61 64 79 0D 0A                                   ady..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:52:57.555021 200.226.10.123:1034 -> xxx.xxx.xxx.xxx:25
TCP TTL:64 TOS:0x0 ID:144  DF
******A* Seq: 0x340DC19B   Ack: 0x2912C6B6   Win: 0x7F23

(... I cut some packets ... and then ... )

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:54:17.421527 200.226.10.123:1034 -> xxx.xxx.xxx.xxx:25
TCP TTL:64 TOS:0x0 ID:159  DF
*****PA* Seq: 0x340DC275   Ack: 0x2912C729   Win: 0x7F88
63 68 61 72 73 65 74 3D 22 22 0D 0A              charset=""..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:54:17.714820 xxx.xxx.xxx.xxx:25 -> 200.226.10.123:1034
TCP TTL:120 TOS:0x0 ID:17769  DF
******A* Seq: 0x2912C729   Ack: 0x340DC281   Win: 0x2152
```

```
(... I cut more packets, and... closing connection . ..)

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:54:42.894836 xxx.xxx.xxx.xxx:25 -> 200.226.10.123:1034
TCP TTL:120 TOS:0x0 ID:44649  DF
***F**A* Seq: 0x2912C749   Ack: 0x340DC2D3   Win: 0x2100

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:54:42.894996 200.226.10.123:1034 -> xxx.xxx.xxx.xxx:25
TCP TTL:64 TOS:0x0 ID:168  DF
******A* Seq: 0x340DC2D3   Ack: 0x2912C74A   Win: 0x7F87

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:54:42.895670 200.226.10.123:1034 -> xxx.xxx.xxx.xxx:25
TCP TTL:64 TOS:0x0 ID:169  DF
***F**A* Seq: 0x340DC2D3   Ack: 0x2912C74A   Win: 0x7F88

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/12-16:54:42.994820 xxx.xxx.xxx.xxx:25 -> 200.226.10.123:1034
TCP TTL:120 TOS:0x0 ID:44905  DF
******A* Seq: 0x2912C74A   Ack: 0x340DC2D4   Win: 0x2100

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

As you can see, it looks like a valid email. But our Snort rule will generate this alarm when this attack happens:

```
[**] Microsoft Exchange 5.5 -  Malformed MIME Header [**]
04/12-16:54:17.421527 200.226.10.123:1034 -> xxx.xxx.xxx.xxx:25
TCP TTL:64 TOS:0x0 ID:159  DF
*****PA* Seq: 0x340DC275   Ack: 0x2912C729   Win: 0x7F88
```

4. Solutions:

As I told before, no router or firewall doing packet filtering could help us in this case. A good solution would be to place a proxy for this service, that would pass to our Exchange servers only valid email messages (Valid in this case means with some valid charset type, not an empty one), but to place it there, we would need some time to implement it, and of course, some cash support (maybe we don't have the machine available). So, the best solution is to apply a Patch to all servers that we have. We know that those patches may take some time to become available, and I see an Alarm like this one from Snort as a good solution to know that we are under attack until those patches are released. (this is better than to wait for somebody call us and say that something is happening to their mail account!)

The official Microsoft patch for this bug is Q248838engI and can be found at:
http://download.microsoft.com/download/exch55/Patch/5.5.2653.22/NT45/EN-US/Q248838engI.EXE

5. Correlations:

Microsoft Security Bulletin (MS00-082) from Oct 30th 2000, available at http://www.microsoft.com/technet/security/bulletin/MS00-082.asp.
Nov 10th, asavelev@eni-net.com posted it at Bugtraq - id 1869 – www.securityfocus.com
I couldn't find any entries or candidates in the Common Vulnerabilities and Exposures because the database version available as the time of this paper is from Oct 13th 2000. http://cve.mitre.org

Assignment 3

Analyze This Scenario

In this assignment we received around 30 files with three types Snort output: alerts, scans and full header dump. We were asked to analyze the traces from this scenario. Unfortunately, we don't have all the data from this network because our sensor was not able to collect data from time to time due to power failures, hard disk problems, etc.

As you can imagine there is high number of events in that files, and we could spend hours looking at them. So I decided to analyze them using the following criteria:

Who is Attacking our network ? How are they attacking ?
Are they internal ?
External ? Coming from where ?
Can we fix or improve defenses ?

Who is Scanning our network ? How are they scanning ?
Are they internal ?
External ? Coming from where ?
Can we fix or improve defenses ?

Do we have some compromised host ?

Attacks:

I decided to group the attacks by types, so we can better read in this paper.
The problem here is that we don't have the rules files from the snort sensor that grab the alerts, so we can't really know why a alarm was trigged. I'm assuming that the rules are standard rules and widely used nowadays.

- wu-ftpd exploit

Here we can see some attempts to exploit a buffer overflow in wu-ftpd. I guess the rule that trigged this alarm was a content-based rule, so the chance that this is a false positive alarm is low. I would recommend to double check that all those servers are running with the latest patches if the are running wu-ftpd. Also, would be nice to look for any scan or reconnaissance coming from that IP in the previous logs.

All the alarms came from 24.17.189.83
hostname = c1032805-c.mckinyl.tx.home.com
@Home Network IP address, a famous source of events.

```
09/08-04:53:17.038845 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:3446-> MY.NET.99.104:21
09/08-05:25:41.092146 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:4640-> MY.NET.150.24:21
09/08-05:25:41.167678 [**] Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:4640-> MY.NET.150.24:21
09/08-05:59:01.961301 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:2362-> MY.NET.202.202:21
09/08-05:59:02.084974 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:2363-> MY.NET.202.190:21
09/08-05:59:04.101862 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:2362-> MY.NET.202.202:21
09/08-05:59:04.191384 [**] Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:2362-> MY.NET.202.202:21
09/08-05:59:04.271433 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 24.17.189.83:2363-> MY.NET.202.190:21
```

Doing a search for wu-ftpd in CVE's database we get around 10 entries and a few candidates. Some of them are: CVE-1999-0080, CVE-1999-0081, CVE-1999-0368, CVE-1999-0955, CAN-1999-0156, CAN-1999-0661, CAN-1999-0911, CAN-2000-0573.

- Happy 99 virus

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

Two sources going to two different email servers.
24.2.2.66 - ha1.rdc1.md.home.com - @Home Network IP address
128.8.198.101 – wmuc.umd.edu – University of Maryland

Another content based rule and the change to be a false positive is low.
An anti-virus program running on a mail-relay or working with our firewall can be used in this case to minimize the impact of this attack.

More information about this worm can be found at http://vil.nai.com/villib/dispVirus.asp?virus_k=10144.

- Tiny Fragments

Source IP addresses:

24.68.58.96	Shaw Fiberlink ltd. - @Home Network
62.76.42.18	mb.ssau.ru - person: Andrei M Sukhov
62.76.42.17	ds.ssau.ru - person: Andrei M Sukhov
213.132.131.201	213-132-131-201.upc.chello.be
212.160.15.85	pa85.konin.ppp.tpnet.pl

There are just a few fragment alarms (10 total) and going to many hosts at different times.
Those packets also could means many things, we can't say what only by looking at the alert message. To really know what they are, we would need to look deeper into the packets. All the comments done in the Detect 4 of this paper are valid for this alerts.

Destination IP addresses: MY.NET.217.82, MY.NET.203.62, MY.NET.1.8, MY.NET.160.109, MY.NET.1.10, MY.NET.1.9, MY.NET.212.86, MY.NET.210.242

Examples:

```
09/11-13:20:45.833385 [**] Tiny Fragments - Possible Hostile Activity [**] 24.68.58.96 -> MY.NET.217.82
09/11-13:21:13.480527 [**] Tiny Fragments - Possible Hostile Activity [**] 24.68.58.96 -> MY.NET.217.82
09/13-16:28:21.252889 [**] Tiny Fragments - Possible Hostile Activity [**] 24.68.58.96 -> MY.NET.210.242
```

- ICQ clients inside our network

Clients: MY.NET.217.42, MY.NET.219.26, MY.NET.222.66, MY.NET.217.82, MY.NET.218.218, MY.NET.220.58, MY.NET.105.2, MY.NET.53.15, MY.NET.206.38
ICQ server - America Online, Inc: 205.188.179.33, 205.188.153.98, 205.188.153.109, 205.188.153.115, 205.188.153.112, 205.188.153.114
This is a good example of False Positive alarms – it's not a Sun RPC high port access attempt.

Some example of alerts (note the source port 4000 at the AOL ICQ server)

```
08/15-04:17:41.894013 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:17:42.081131 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
```

```
08/15-04:17:43.935275 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:17:46.918388 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:18:40.061661 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:19:40.058912 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:20:40.070093 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:21:40.049280 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:22:40.013677 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:23:40.059560 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
08/15-04:24:40.006273 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000-> MY.NET.217.42:32771
```

Not ICQ servers
141.213.191.50 – westquas-191-50.reshall.umich.edu - University of Michigan
128.211.224.100 – wily-a-100.resnet.purdue.edu - Purdue University

Those ones can be Sun RPC high port access attempt!

```
08/20-22:57:10.153306 [**] Attempted Sun RPC high port access [**] 128.211.224.100:2917-> MY.NET.98.111:32771
09/12-18:20:59.295476 [**] Attempted Sun RPC high port access [**] 141.213.191.50:3787-> MY.NET.98.160:32771
09/12-18:58:28.164729 [**] Attempted Sun RPC high port access [**] 141.213.191.50:4670-> MY.NET.98.160:32771
09/12-19:14:42.174099 [**] Attempted Sun RPC high port access [**] 141.213.191.50:4090-> MY.NET.98.160:32771
```

- External RPC Calls:

Portmapper is a service that keeps a directory of all the RPC services running on that machine. Here, we can see some queries to it. Probably the attacker was trying to find out what services are running and in what port.

```
08/18-06:10:13.484691 [**] External RPC call [**] 18.116.0.75:111-> MY.NET.6.15:111
08/18-06:10:13.546328 [**] External RPC call [**] 18.116.0.75:693-> MY.NET.6.15:111
08/18-06:10:13.561808 [**] External RPC call [**] 18.116.0.75:693-> MY.NET.6.15:111
08/18-06:10:13.561937 [**] External RPC call [**] 18.116.0.75:693-> MY.NET.6.15:111
08/18-06:10:13.603288 [**] External RPC call [**] 18.116.0.75:693-> MY.NET.6.15:111
08/18-06:10:13.603333 [**] External RPC call [**] 18.116.0.75:693-> MY.NET.6.15:111
08/19-10:11:34.624674 [**] External RPC call [**] 209.160.238.215:2572-> MY.NET.6.15:111
08/19-10:11:47.587415 [**] External RPC call [**] 209.160.238.215:4980-> MY.NET.15.127:111
08/19-10:13:58.565702 [**] External RPC call [**] 209.160.238.215:2815-> MY.NET.100.130:111
```

There is one thing about this log that can make us look again to it. Portmapper queries are not big and normally don't need many packets. We can see some high activity from 18.116.0.75 to the same host. This could be an attempt to insert or maybe delete a service. We should start watching better for port 111.

The IP addresses doing this attack are:

18.116.0.75	FLUTTER.MIT.EDU
209.160.238.215	frankenstein.nwnii.com
141.223.124.31	Pohang Institute of Science and Technology - Korea
161.31.208.237	University of Central Arkansas – US
210.100.199.219	Korea Telecom
210.101.101.110	Korea Telecom

The top target host for this attack is MY.NET.6.15!

- SMB Name Wildcard

We could see many alerts with this pattern. Here are some examples:

```
08/11-16:38:04.318402 [**] SMB Name Wildcard [**] 166.72.86.217:137-> MY.NET.100.165:137
08/11-16:38:40.624689 [**] SMB Name Wildcard [**] 166.72.86.217:137-> MY.NET.100.165:137
08/11-16:38:41.032245 [**] SMB Name Wildcard [**] 166.72.86.217:137-> MY.NET.100.165:137
08/11-16:38:42.143327 [**] SMB Name Wildcard [**] 166.72.86.217:137-> MY.NET.100.165:137
08/11-16:26:04.299200 [**] SMB Name Wildcard [**] 62.136.168.18:1124-> MY.NET.70.121:137
08/11-16:26:23.147585 [**] SMB Name Wildcard [**] 62.136.168.18:1124-> MY.NET.70.121:137
08/11-16:13:19.788046 [**] SMB Name Wildcard [**] 205.229.90.194:63733-> MY.NET.181.37:137
08/11-16:51:17.369342 [**] SMB Name Wildcard [**] 24.28.62.226:1975-> MY.NET.70.121:137
```

NetBIOS, everyday we learn something new about it. There is always a new scenario where the default behavior changes and we get different alarms. In this log we can see basically 2 types of messages. In the first one the connections are coming from some high port numbers and in the second one, the source port is 137. Source port 137 going to port 137 indicates that the source host is running samba. High port source can be a normal behavior of many servers, like an Exchange server trying to resolve names after it receives a message. So not always those alerts are true. Many times we get false positives. Port 137 is a great source of information for an attacker and I suggest to block it whenever is possible.

The top source IP addresses of this attack are:

MY.NET.101.160	Hey, an internal host! Maybe it is compromised… Let's keep an eye on him.
166.72.86.217	slip166-72-86-217.ma.us.prserv.net
131.118.254.222	umcp-222.dialup.umbc.edu
168.143.29.9	gauntlete.nga.gov
207.79.66.3	slip129-37-161-200.on.ca.prserv.net

- SNMP public access

We can see many activities from our internal hosts probably responding to a query from a management station (MY.NET.101.192) with the public SNMP community string. We should change that to something hard to guess because SNMP can be a wonderful source ofinformation for an attacker.

Examples of Alerts:

```
08/15-19:59:52.855020 [**] SNMP public access [**] MY.NET.97.154:1049-> MY.NET.101.192:161
09/03-12:42:35.351628 [**] SNMP public access [**] MY.NET.98.109:1045-> MY.NET.101.192:161
09/03-19:23:50.069324 [**] SNMP public access [**] MY.NET.98.114:1063-> MY.NET.101.192:161
09/10-15:27:45.937458 [**] SNMP public access [**] MY.NET.98.172:1042-> MY.NET.101.192:161
09/11-18:31:00.333609 [**] SNMP public access [**] MY.NET.97.217:1066-> MY.NET.101.192:161
```

The hosts that we should review first are:

MY.NET.98.172
MY.NET.98.109
MY.NET.97.217
MY.NET.97.154
MY.NET.98.114
MY.NET.98.171
MY.NET.98.191
MY.NET.98.181
MY.NET.97.244
MY.NET.98.159
MY.NET.98.201
MY.NET.98.148
MY.NET.97.246
MY.NET.97.206
MY.NET.98.177
MY.NET.98.190

- Wingate 1080 attempt

We could see many alerts regarding WinGate proxy servers. Some of them were only scans but some really look like valid traffic. WinGate proxy servers can be used to forward attacks to other machines and we cannot allow that.

IP doing scans for WinGate servers:

168.187.26.157	Kuwait Ministry of Communications
168.120.16.250	Assumption University – Thailand

Most activities are hitting our host MY.NET.60.11 at port 1080, and that makes me believe we have a running proxy server at that host. We better check it and maybe change his IP address and hostname.

```
09/10-11:44:04.264653 [**] WinGate 1080 Attempt [**] 208.194.161.50:4631-> MY.NET.60.11:1080
09/11-13:32:13.830778 [**] WinGate 1080 Attempt [**] 209.10.218.250:3081-> MY.NET.60.11:1080
09/11-14:03:58.931557 [**] WinGate 1080 Attempt [**] 216.152.64.130:60114-> MY.NET.60.11:1080
09/12-12:29:10.530372 [**] WinGate 1080 Attempt [**] 207.126.106.118:4731-> MY.NET.60.11:1080
09/12-12:29:11.749308 [**] WinGate 1080 Attempt [**] 216.67.82.51:1482-> MY.NET.60.11:1080
09/12-12:54:53.981779 [**] WinGate 1080 Attempt [**] 63.103.51.71:2465-> MY.NET.60.11:1080
09/12-12:56:49.493771 [**] WinGate 1080 Attempt [**] 216.67.82.51:1590-> MY.NET.60.11:1080
09/12-12:56:49.919182 [**] WinGate 1080 Attempt [**] 63.103.51.71:2469-> MY.NET.60.11:1080
```

There are a few entries at http://cve.mitre.org regarding WinGate: CVE-1999-0290, CVE-1999-0291, CVE-1999-0441, CVE-1999-0494, CAN-1999-0657.

- Napster users inside our network

Napster users consume as much bandwidth as you give them. Maybe we should block it.

Examples of alerts of this type:

```
08/17-12:50:46.184153 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.2:4807-> MY.NET.181.87:6699
09/09-10:45:10.461542 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.2:22756-> MY.NET.221.94:6699
09/14-07:45:06.948211 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.174:2173-> MY.NET.157.200:6699
09/14-07:45:07.375438 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.174:2173-> MY.NET.157.200:6699
```

Also, activities from port 7777 were found, mainly from 63.248.55.245 - Flashcom, Inc - USA

SCANS

- Nmap finger print attempt

We can see in many scans a fingerprint identification attempt. This reconnaissance is based on the response get from the server to many different flags combinations and the 2 reserved bits from the TCP header. Different OS respond differently! There are even some vendors not using the RFP specification. The top scanners are:

24.180.134.156	cc349491-a.hwrd1.md.home.com
130.149.41.70	bessy.physik.TU-Berlin.de
207.151.147.201	pool.207.151.147.201.cinenet.net
151.196.73.119	csl.windermeregroup.com
63.226.208.41	aazpppdsl41.sttl.uswest.net

Examples:

```
09/11-04:48:56.731170 [**] Probable NMAP fingerprint attempt [**] 24.180.134.156:50111-> MY.NET.208.5:23
09/11-04:49:52.852370 [**] Probable NMAP fingerprint attempt [**] 24.180.134.156:50111-> MY.NET.208.17:23
```

A really good document explaining how nmap fingerprint is done can be found at: http://www.insecure.org/nmap/nmap-fingerprinting-article.html

- Queso fingerprint:

Another program to identify OS remotely. More info can be found at: http://www.apostols.org/projectz/queso/

Lots of Alerts in our files…

```
09/08-20:14:18.371606 [**] Queso fingerprint [**] 64.80.63.121:3240-> MY.NET.210.194:113
09/08-20:28:29.912367 [**] Queso fingerprint [**] 64.80.63.121:3856-> MY.NET.210.194:113
09/10-10:29:43.642064 [**] Queso fingerprint [**] 64.80.63.121:3360-> MY.NET.223.42:113
09/10-10:29:46.908037 [**] Queso fingerprint [**] 64.80.63.121:3364-> MY.NET.223.42:113
```

- Looking for hosts with Telnet port open

Not only useful for brute force attacks and password guessing, but also several attacks against telnet server are know. Like, CVE-2000-0268 (Cisco IOS 11.x and 12.x denial of service), CVE-1999-0073 (user can set environment variables to bypass the normal values and gain root access) or CVE-1999-0273 (Denial of service through Solaris 2.5.1).

Method utilized by this IP: SYN/FYN scan

210.101.101.110	Korea Telecom - Seoul, Korea - Person: Gisu Choi - Phone +82 2 766 1407
24.180.134.156	@Home Network IP Address (also doing OS finger print)

Method utilized by this IP: SYN scan

Pretty Aggressive Scans:

129.186.93.133	skinner.cs.iastate.edu
128.171.57.194	micro194.lib3.hawaii.edu
24.180.134.156	cc349491-a.hwrd1.md.home.com

Not so heavy (maybe better target)

216.99.200.242	securedesign.net
216.138.44.49	as52-49.adams.net
207.151.147.201	pool.207.151.147.201.cinenet.net
210.56.17.202	AS5300.PRI3.13.Dialup.Lahore.comsats.net.pk

Top Targets
MY.NET.20.219
MY.NET.208.166
MY.NET.17.245

Examples:

```
Sep 2 10:14:17 210.101.101.110:23-> MY.NET.6.15:23 SYNFIN **SF****
Sep 11 04:48:14 24.180.134.156:50109-> MY.NET.208.1:23 SYN 2*S***** RESERVEDBITS
```

- Looking for hosts with FTP port open

Some attacks related to ftp:
CAN-2000-0573 - wu-ftpd 2.6.0 - allows remote attackers to execute arbitrary commands via the SITE EXEC command.
CVE-2000-0636 - HP JetDirect printers versions denial of service via a malformed FTP quote command.
CVE-1999-0777 - IIS FTP servers may allow a remote attacker to read or delete files on the server, even if they have "NoAccess" permissions.

Large Scans: hitting almost every host in MY.NET.x.y/16

195.114.226.41	apollo-dh0040.multiweb.net - Method: SYN scan
212.141.100.97	gw2a61-1-d97.wind.it – Method: SYN scan
194.165.230.250	ume-gw.resonia.se – Method: SYN scan

Few Hosts on multiple subnets

213.188.8.45	albatross.fast.no
4.54.37.160	PPPa67-City-4R7146.saturn.bbn.com
24.94.176.113	mkc-94-176-113.kc.rr.com
206.18.105.224	p32-max2.akl.ihugultra.co.nz
207.151.147.201	pool.207.151.147.201.cinenet.net
210.61.144.125	Abnet Information Co., Ltd - TW

- Trojans Scans

Most scanned Trojan: SubSeven BackDoor Scan: Method SYN Scan
http://subseven.slak.org/
Old versions of SubSeven server, by default bind to the port 27374, so it is pretty common to see those scan nowadays. The latest version of Subseven uses port 6667 as the default, but of course this port can be easily changed.

Top Scanner: 35.10.82.111 – Scanned all host on MY.NET.x.y/16 in 40 minutes.

```
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*****
Aug 16 04:35:20 35.10.82.111:2816 -> MY.NET.1.8:27374 SYN **S*****
Aug 16 04:35:20 35.10.82.111:2817 -> MY.NET.1.9:27374 SYN **S*****
Aug 16 04:35:20 35.10.82.111:2818 -> MY.NET.1.10:27374 SYN **S*****
```

- Top Port Scanners

We had some pretty large port scans in our network. The most hit hosts are MY.NET.60.8, MY.NET.217.10, MY.NET.150.89, MY.NET.20.10 and MY.NET.60.7

Top Sources of TCP scans:

207.151.147.201	pool.207.151.147.201.cinenet.net – CA, USA
195.57.243.171	Internet Service Provider – Spain

134.28.9.225	Technische Universitaet Hamburg-Harburg, DE

Examples:

```
Aug 16 01:41:05 207.151.147.201:4066-> MY.NET.60.8:192 SYN **S*****
Aug 16 01:41:05 207.151.147.201:4070-> MY.NET.60.8:120 SYN **S*****
Aug 16 01:41:05 207.151.147.201:4078-> MY.NET.60.8:4444 SYN **S*****
Aug 16 01:41:05 207.151.147.201:4079-> MY.NET.60.8:1472 SYN **S*****
Aug 17 13:55:01 147.208.171.139:4326-> MY.NET.150.89:1090 SYN **S*****
Aug 17 13:55:01 147.208.171.139:4330-> MY.NET.150.89:1243 SYN **S*****
Aug 17 13:55:02 147.208.171.139:4368-> MY.NET.150.89:2583 SYN **S*****
Aug 17 13:55:02 147.208.171.139:4377-> MY.NET.150.89:2801 SYN **S*****
```

147.208.171.139 – security.Norton.com – Also did some scans, but this is a server that checks for common vulnerabilities remotely – false positive alert. What happened here, is that our hosts like MY.NET.150.89 probably asked this free check in the Norton.com website.

213.25.136.60 - server.roztocze.com.pl was doing a heavy SYN/FYN scan in a large number of host on TCP port 9704 (not a common port).

- Strange Behavior - Probably compromised

This can be a response from MY.NET.1.3 to some DNS query from our hosts because all the packets are going to high ports and almost never to a port that have some famous Trojan or Backdoor program. If we don't have MY.NET.1.3 as a DNS server, then I would be worried about this log.

```
Sep 3 09:07:09 MY.NET.1.3:53-> MY.NET.100.187:1029 UDP
Sep 3 09:07:09 MY.NET.1.3:53-> MY.NET.179.52:1814 UDP
Sep 3 09:07:09 MY.NET.1.3:53-> MY.NET.162.198:4371 UDP
Sep 3 09:07:09 MY.NET.1.3:53-> MY.NET.104.18:1311 UDP
```

Also, we can see some NTP packets from MY.NET.1.3 going to a few hosts on our network. Make sure this is the normal behavior of this host.

```
Sep 3 09:10:26 MY.NET.1.3:123-> MY.NET.100.37:123 UDP
Sep 3 09:10:26 MY.NET.1.3:123-> MY.NET.139.121:123 UDP
Sep 3 09:10:27 MY.NET.1.3:123-> MY.NET.60.75:123 UDP
```

The same happens to hosts MY.NET.1.4, MY.NET.1.5 and MY.NET.1.13. Also MY.NET.217.218 and MY.NET.223.14 were the source of pretty strange logs.

Assignment 4

To do the previous assignment we received about 60 files with Snort output. As we said before, they were not sequential or complete. The first thing I decided to do was to organize them into 3 big files, each one of a kind, without duplicated lines. We had files like SnortS20 and SnortS21 that were exactly the same (I even did a *diff file1 file2* to make sure).

After that, I used some poor home made perl scripts to take a quickly look at the alerts and scans and see what was going on. To do that, I had to change the string "MY.NET" to some number. I choose to use 10.10 since there was no reference to it in the files. Using the output from those scripts I was able to cut some lines of each file, so I could reduce a little bit the size of each file. What I took out was some pretty heavy network mapping and things like that. I did that because I was going to use SnortSnarf to help me in the analysis process http://www.silicondefense.com/snortsnarf/ and I was going to use a Pentium II 233 with only 64Mb RAM to analyze and create the .html so I needed smaller files. Actually, I did it in the other way… After I got some Out of Memory messages and swap problems I decided to do it! (Of course, I didn't discard that info I just analyzed it apart).

Also, I imported some files into Microsoft Excel so I could organize the lines for a better analysis, like putting some Source IP together or looking for port xxx. But here goes a suggestion for you guys that are going to do your practical now, listen to folks like Lenny Zeltser that saved his info into Berkeley Databases http://www.sans.org/y2k/practical/Lenny_Zeltser.htm it is much better! Excel has many problems dealing with thousand of lines in each spreadsheet and you are going to regret using it, just like I did.

After I had the output of SnortSnarf and played enough with Excel, I went back to my scripts, awk and grep commands to correlate events, alarms and scans.