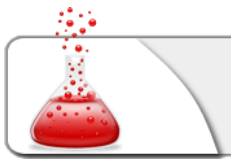
Traffic Analysis Challenges



LAB 1

LAB



1. Scenario

You have been tasked by your SOC manager to analyze various traffic capture files and answer his questions. Each task is independent and not connected with the other ones in any way.

2. Learning Objectives

- Refresh your networking knowledge
- Learn to identify TCP spoofing and internal botnet-like activity
- Practice identifying attacks by analyzing network traffic, including IPv6-based ones

3. Recommended tools

- Wireshark / tcpdump
- NetworkMiner

4. Tasks

Task 1: Analyze the three provided pcap files and identify if the web server was actually attacked and how

An Internet-facing production web server (internal IP: **192.168.1.6**) experienced serious performance degradation. The SOC manager provided you with three (3) pcap files.

normal_webserver_traffic_1.pcap: Contains normal web server traffic

normal_webserver_traffic_2.pcap: Contains normal web server traffic as well



webserver_traffic.pcap: Contains traffic captured at the web server host, at the time of the incident

Use the knowledge you obtained so far to identify if your organization is dealing with a real incident as well as the attacker TTPs, if this is the case.

Note: ARP poisoning defenses exist in the intranet. Any ARP packet can be considered authentic.

TASK 2: ANALYZE THE PROVIDED DATA AND IDENTIFY GEOGRAPHICALLY WHERE THE MAN-IN-THE-MIDDLE ATTACK TOOK PLACE

The Beijing-based office reported a man-in-the-middle attack when trying to access the company's repositories on github.com.

They identified this was the case when they were presented with an untrusted SSL certificate while trying to browse the company's repositories.

The SOC manager provided you with a pcap file and a *traceroute* command output.

employee_traffic.pcap: Contains traffic captured at an employee's machine at the time of the attack.

The *traceroute* command, whose output you see below, was execute at the Beijing office.

```
traceroute to 207.97.227.239 (207.97.227.239), 30 hops max, 38 byte packets
1 gw (202.38.128.2) 0.231 ms 0.175 ms 0.178 ms
2 202.122.36.1 (202.122.36.1) 0.719 ms 0.626 ms 0.652 ms
3 192.168.1.25 (192.168.1.25) 0.509 ms 0.485 ms 0.459 ms
4 8.131 (159.226.253.77) 0.516 ms 0.488 ms 0.496 ms
5 8.198 (159.226.253.54) 11.981 ms 0.913 ms 0.829 ms
6 8.192 (159.226.254.254) 40.477 ms 40.614 ms 40.524 ms
7 Gi6-0.gw2.hkg3.asianetcom.net (203.192.137.173) 53.062 ms 40.610 ms 40.617 ms
8 te0-3-0-0.wr1.hkg0.asianetcom.net (61.14.157.249) 42.459 ms 43.133 ms 43.765 ms
9 te0-1-0-0.wr1.osa0.asianetcom.net (61.14.157.58) 76.603 ms 76.586 ms 76.648 ms
10 gi6-0-0.gw1.sjc1.asianetcom.net (61.14.157.98) 195.939 ms 195.934 ms 196.090 ms
11 ip-202-147-50-250.asianetcom.net (202.147.50.250) 180.966 ms 181.058 ms 181.339 ms
12 dcx2-edge-01.inet.qwest.net (67.14.28.70) 342.431 ms 340.882 ms 338.639 ms
```



```
13 67.133.246.158 (67.133.246.158) 253.975 ms 253.407 ms 253.742 ms
14 vlan905.core5.iad2.rackspace.net (72.4.122.10) 253.468 ms 252.947 ms 253.639 ms
15 aggr301a-1-core5.iad2.rackspace.net (72.4.122.121) 253.767 ms 253.862 ms 253.010 ms
16 github.com (207.97.227.239) 254.104 ms 253.789 ms 253.638 ms
```

Taking into consideration the provided data above, identify in which country/part of the route the man-in-the-middle attack was executed.

Note: Try working with NetworkMiner this time.

This challenge was inspired by a real man-in-the-middle attack against Github and the attack's analysis post by Erik Hjelmvik.

TASK 3: ANALYZE THE PROVIDED CAP FILE AND CONCLUDE WHETHER AN INCIDENT SHOULD BE DECLARED OR NOT

The SOC manager provided you with the **intranet.cap** file, which contains some suspicious traffic. The attack was identified when a host started experiencing connectivity issues.

Notes:

1. The cap file was sanitized and contains only the suspicious portion of the captured traffic
2. The NOC team indicated that frames #1 and #2 are legitimate



SOLUTIONS

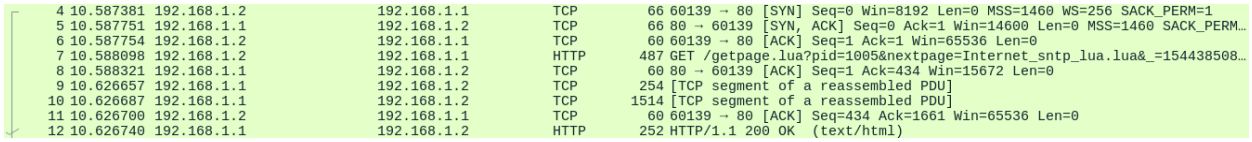


Below, you can find solutions for each task for the Network Security, Exploitation tasks. Remember though, that you can follow your own strategy (which may be different from the one explained in the following lab).

TASK 1: ANALYZE THE THREE PROVIDED PCAP FILES AND IDENTIFY IF THE WEB SERVER WAS ACTUALLY ATTACKED AND HOW

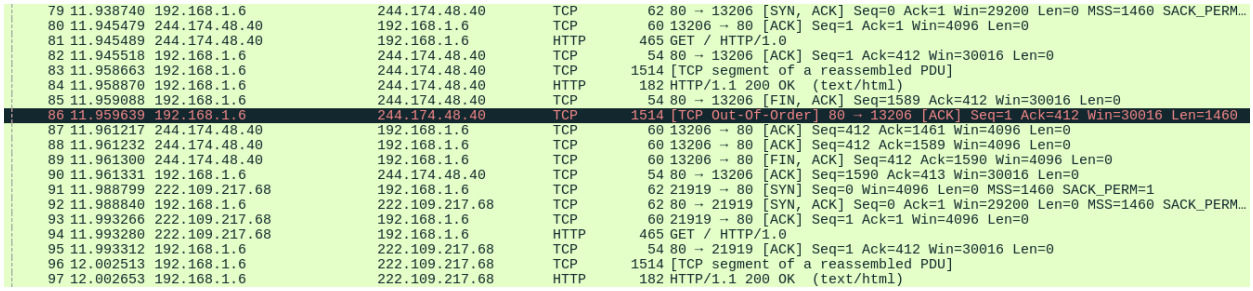
Start by analyzing the **normal_webserver_traffic_1.pcap** and **normal_webserver_traffic_2.pcap** files, to get familiar with how normal traffic looks like in the case of this server.

Those two pcap files depict normal interaction between clients and a web server. This is some initial TCP messages to establish communication and some HTTP traffic afterward.



4	10.587381	192.168.1.2	192.168.1.1	TCP	66 60139 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	10.587751	192.168.1.1	192.168.1.2	TCP	66 80 → 60139 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM…
6	10.587754	192.168.1.2	192.168.1.1	TCP	60 60139 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
7	10.588098	192.168.1.2	192.168.1.1	HTTP	487 GET /getpage.lua?pid=1005&nextpage=Internet_sntp_lua.lua&_=154438508…
8	10.588321	192.168.1.1	192.168.1.2	TCP	60 80 → 60139 [ACK] Seq=1 Ack=434 Win=15672 Len=0
9	10.626657	192.168.1.1	192.168.1.2	TCP	254 [TCP segment of a reassembled PDU]
10	10.626687	192.168.1.1	192.168.1.2	TCP	1514 [TCP segment of a reassembled PDU]
11	10.626700	192.168.1.2	192.168.1.1	TCP	60 60139 → 80 [ACK] Seq=434 Ack=1661 Win=65536 Len=0
12	10.626740	192.168.1.1	192.168.1.2	HTTP	252 HTTP/1.1 200 OK (text/html)

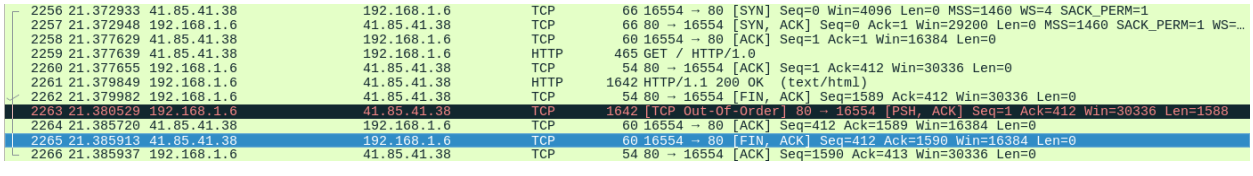
Now, load **webserver_traffic.pcap** into Wireshark.



79	11.938740	192.168.1.6	244.174.48.40	TCP	62 80 → 13206 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM…
80	11.945479	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
81	11.945489	244.174.48.40	192.168.1.6	HTTP	465 GET / HTTP/1.0
82	11.945518	192.168.1.6	244.174.48.40	TCP	54 80 → 13206 [ACK] Seq=1 Ack=412 Win=30016 Len=0
83	11.958663	192.168.1.6	244.174.48.40	TCP	1514 [TCP segment of a reassembled PDU]
84	11.958870	192.168.1.6	244.174.48.40	HTTP	182 HTTP/1.1 200 OK (text/html)
85	11.959088	192.168.1.6	244.174.48.40	TCP	54 80 → 13206 [FIN, ACK] Seq=1589 Ack=412 Win=30016 Len=0
86	11.959639	192.168.1.6	244.174.48.40	TCP	1514 [TCP Out-Of-Order] 80 → 13206 [ACK] Seq=1 Ack=412 Win=30016 Len=1460
87	11.961217	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [ACK] Seq=412 Ack=1461 Win=4096 Len=0
88	11.961232	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [ACK] Seq=412 Ack=1589 Win=4096 Len=0
89	11.961300	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [FIN, ACK] Seq=412 Ack=1590 Win=4096 Len=0
90	11.961331	192.168.1.6	244.174.48.40	TCP	54 80 → 13206 [ACK] Seq=1590 Ack=413 Win=30016 Len=0
91	11.988799	222.109.217.68	192.168.1.6	TCP	62 21919 → 80 [SYN] Seq=0 Win=4096 Len=0 MSS=1460 SACK_PERM=1
92	11.988840	192.168.1.6	222.109.217.68	TCP	62 80 → 21919 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM…
93	11.993266	222.109.217.68	192.168.1.6	TCP	60 21919 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
94	11.993280	222.109.217.68	192.168.1.6	HTTP	465 GET / HTTP/1.0
95	11.993312	192.168.1.6	222.109.217.68	TCP	54 80 → 21919 [ACK] Seq=1 Ack=412 Win=30016 Len=0
96	12.002513	192.168.1.6	222.109.217.68	TCP	1514 [TCP segment of a reassembled PDU]
97	12.002653	192.168.1.6	222.109.217.68	HTTP	182 HTTP/1.1 200 OK (text/html)

We see multiple clients requesting web server resources over HTTP. For sure, the amount of requests is big, and requests are being sent at short intervals, but this a production web server, so, heavy traffic is **probably** normal. Let's dive in a little deeper by focusing only on one client, for example, the **41.85.41.38** one.

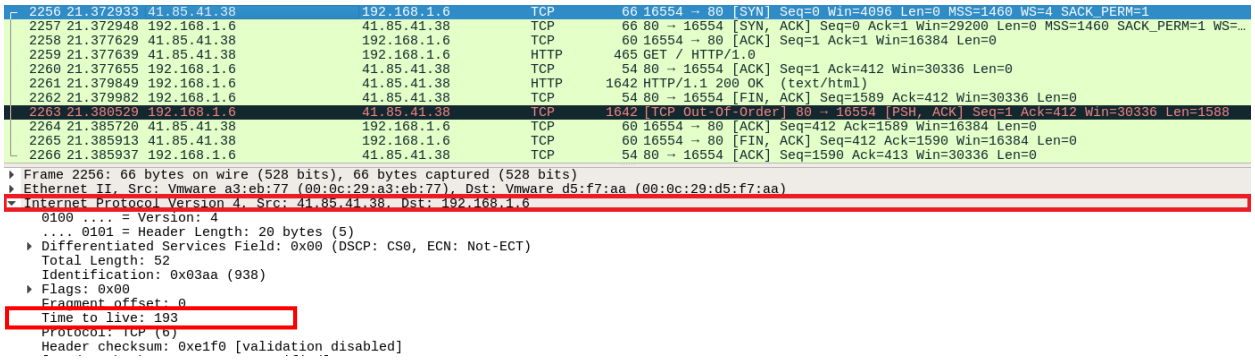
```
ip.addr==41.85.41.38
```

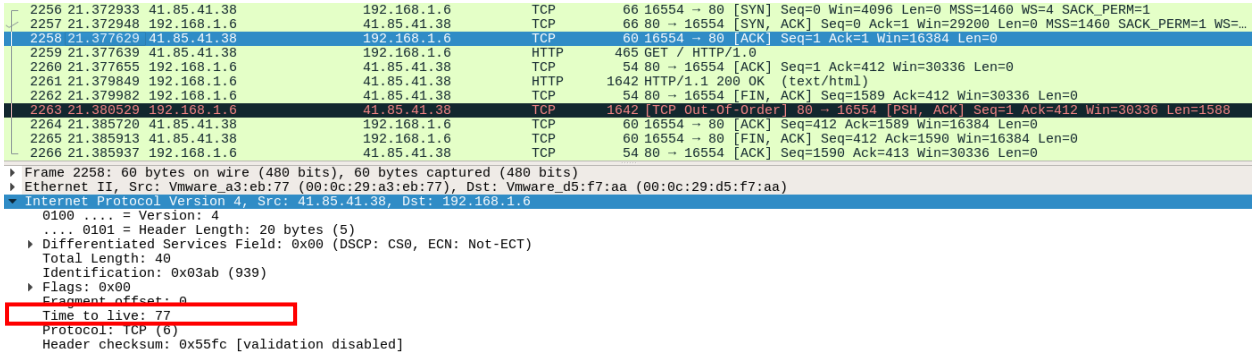


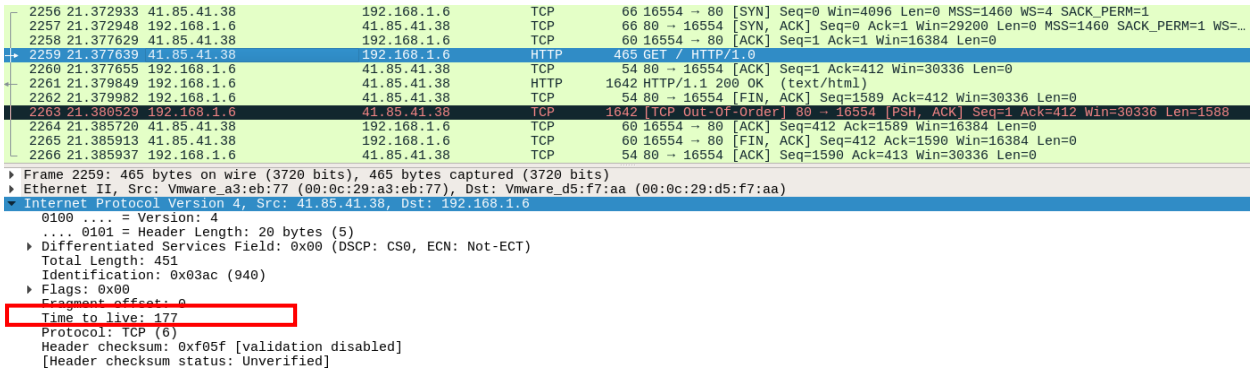
2256	21.372933	41.85.41.38	192.168.1.6	TCP	66 16554 → 80 [SYN] Seq=0 Win=4096 Len=0 MSS=1460 WS=4 SACK_PERM=1
2257	21.372948	192.168.1.6	41.85.41.38	TCP	66 80 → 16554 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=…
2258	21.377629	41.85.41.38	192.168.1.6	TCP	60 16554 → 80 [ACK] Seq=1 Ack=1 Win=16384 Len=0
2259	21.377639	41.85.41.38	192.168.1.6	HTTP	465 GET / HTTP/1.0
2260	21.377655	192.168.1.6	41.85.41.38	TCP	54 80 → 16554 [ACK] Seq=1 Ack=412 Win=30336 Len=0
2261	21.379849	192.168.1.6	41.85.41.38	HTTP	1642 HTTP/1.1 200 OK (text/html)
2262	21.379982	192.168.1.6	41.85.41.38	TCP	54 80 → 16554 [FIN, ACK] Seq=1589 Ack=412 Win=30336 Len=0
2263	21.380529	192.168.1.6	41.85.41.38	TCP	1642 [TCP Out-Of-Order] 80 → 16554 [PSH, ACK] Seq=1 Ack=412 Win=30336 Len=1588
2264	21.385720	41.85.41.38	192.168.1.6	TCP	60 16554 → 80 [ACK] Seq=412 Ack=1589 Win=16384 Len=0
2265	21.385913	41.85.41.38	192.168.1.6	TCP	60 16554 → 80 [FIN, ACK] Seq=412 Ack=1590 Win=16384 Len=0
2266	21.385937	192.168.1.6	41.85.41.38	TCP	54 80 → 16554 [ACK] Seq=1590 Ack=413 Win=30336 Len=0



Now, expand the Internet Protocol in Wireshark and go through each packet that has **41.85.41.38** as Source.







If you look at those packets carefully, you will notice abnormal TTL value fluctuations. Of course, route(s) may change over time and cause (slight) TTL fluctuations, but the fluctuations we see on this traffic capture are too wide to be justified.

This behavior is very odd for an **external** client – web server communication.



Perform the same steps against the **normal_webserver_traffic_1.pcap** and **normal_webserver_traffic_2.pcap** capture files, and you will see that there are no TTL fluctuations.

Also try filtering the whole **webserver_traffic.pcap** capture file based on the **41.85.41.38** client's MAC address.

```
eth.addr==00:0c:29:a3:eb:77
```

2	3.193290	7c:39:53:8e:e5:60	Vmware_a3:eb:77	ARP	60 Who has 192.168.1.4? Tell 192.168.1.1
3	3.193440	Vmware_a3:eb:77	7c:39:53:8e:e5:60	ARP	60 192.168.1.4 is at 00:0c:29:a3:eb:77

The first thing you should notice is that **00:0c:29:a3:eb:77** belongs to an intranet machine, specifically the **192.168.1.4** one. Remember, that the intranet features ARP spoofing defenses, so, this message is authentic.

62	10.241559	192.168.1.4	54.187.16.16	TCP	60 50334 → 443 [RST] Seq=359 Win=0 Len=0
65	11.003461	192.168.1.4	93.184.220.29	TCP	66 48260 → 80 [FIN, ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2335570465 T…
66	11.087836	93.184.220.29	192.168.1.4	TCP	66 80 → 48260 [FIN, ACK] Seq=1 Ack=2 Win=144896 Len=0 TSval=2419931124 …
67	11.088011	192.168.1.4	93.184.220.29	TCP	66 48260 → 80 [ACK] Seq=2 Ack=2 Win=29312 Len=0 TSval=2335570550 TSecr=…
78	11.938683	244.174.48.40	192.168.1.6	TCP	62 13206 → 80 [SYN] Seq=0 Win=4096 Len=0 MSS=1460 SACK_PERM=1
80	11.945479	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
81	11.945489	244.174.48.40	192.168.1.6	HTTP	465 GET / HTTP/1.0
87	11.961217	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [ACK] Seq=412 Ack=1461 Win=4096 Len=0
88	11.961232	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [ACK] Seq=412 Ack=1589 Win=4096 Len=0
89	11.961300	244.174.48.40	192.168.1.6	TCP	60 13206 → 80 [FIN, ACK] Seq=412 Ack=1590 Win=4096 Len=0
91	11.988799	222.109.217.68	192.168.1.6	TCP	62 21919 → 80 [SYN] Seq=0 Win=4096 Len=0 MSS=1460 SACK_PERM=1
93	11.993266	222.109.217.68	192.168.1.6	TCP	60 21919 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
94	11.993280	222.109.217.68	192.168.1.6	HTTP	465 GET / HTTP/1.0
100	12.009628	222.109.217.68	192.168.1.6	TCP	60 21919 → 80 [ACK] Seq=412 Ack=1589 Win=4096 Len=0
101	12.009649	222.109.217.68	192.168.1.6	TCP	60 21919 → 80 [FIN, ACK] Seq=412 Ack=1590 Win=4096 Len=0
103	12.038911	214.221.36.5	192.168.1.6	TCP	74 31366 → 80 [SYN] Seq=0 Win=4096 Len=0 WS=1024 MSS=265 TSval=10611095…

Not only that but **00:0c:29:a3:eb:77** is the MAC address of every client in this capture file!

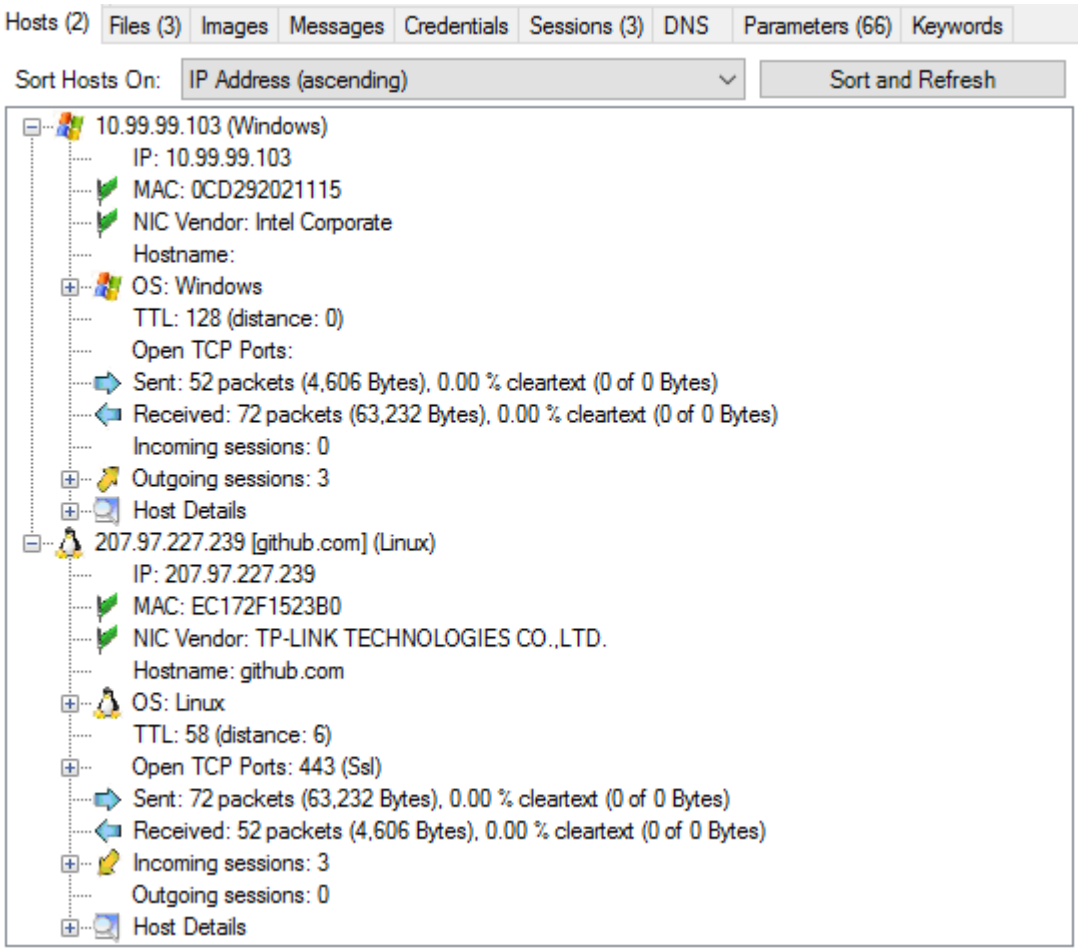
To sum up, we identified so far that some heavy traffic is sent to the internet-facing production web server (internal IP: **192.168.1.6**), by an **intranet** machine (**192.168.1.4**) but the traffic was ***crafted*** to seem like it originates from multiple different (external) hosts.

If you perform some research online, you will find out that such wide TTL fluctuations can be part of a TCP (Source) spoofing routine. Based on what we have identified above and the amount of requests that were being sent (at short intervals), we can conclude that the **192.168.1.4** intranet machine was trying to cause DoS on the **192.168.1.6** web server, simulating botnet-like traffic. Botnet-like traffic facilitates obscuring the source of an attack and also makes incident response harder.



TASK 2: ANALYZE THE PROVIDED DATA AND IDENTIFY GEOGRAPHICALLY WHERE THE MAN-IN-THE-MIDDLE ATTACK TOOK PLACE

Start by loading the **employee_traffic.pcap** into NetworkMiner. You should see the following:



Before starting the analysis, browse the **Files** tab of NetworkMiner. There, you will find the presented and invalid SSL certificates. Remember this NetworkMiner feature, since it is quite handy during Incident Response activities.



The first thing to notice is the TTL (58) and TCP Window Size (5840) of the Github server. If you recall, IP initial TTL and TCP Window Size can be used for Passive OS Fingerprinting activities.

Table 1. Popular OSs' time to live (TTL) and window size values.		
OS	**TTL**	**Window size (bytes)**
Linux 2.4 and 2.6	64	5,840
Google customized Linux	64	5,720
Linux kernel 2.2	64	32,120
FreeBSD	64	65,535
OpenBSD, AIX 4.3	64	16,384
Windows 2000	128	16,384
Windows XP	128	65,535
Windows 7, Vista, and Server 8	128	8,192
Cisco Router IOS 12.4	255	4,128
Solaris 7	255	8,760
MAC	64	65,535

That being said, the Github server seems to be a Linux-based one. NetworkMiner also suggests the same. A TTL of 58 means that the SSL connection was terminated six hops away from the user. NetworkMiner states the same (distance: 6).

Now, focus on the *traceroute* command output. Six hops away in the *traceroute* command output you see the following.

```
8.192 (159.226.254.254) 40.477 ms 40.614 ms 40.524 ms
```

If you use a geolocation service for IPs, such as https://www.iplocation.net/, and search for this IP address, you will see the below.

Geolocation data from IP2Location (Product: DB6, updated on 2018-11-1)

IP Address	Country	Region	City
159.226.254.254	China	Beijing	Mentougou
ISP	**Organization**	**Latitude**	**Longitude**
China Science and Technology Network	Not Available	39.9392	116.0917



The above means that the man-in-the-middle attack was most probably executed within the Beijing region and not within another region or, even worse, another country.

TASK 3: ANALYZE THE PROVIDED CAP FILE AND CONCLUDE WHETHER AN INCIDENT SHOULD BE DECLARED OR NOT

Start by loading **intranet.cap** into Wireshark. You should see the below.

1	0.000000	fe80::2	ff02::1:ff00:1	ICMPv6	86 Neighbor Solicitation for fe80::1 from 00:00:00:00:00:0b
2	0.005346	fe80::1	fe80::2	ICMPv6	86 Neighbor Advertisement fe80::1 (sol, ovr) is at 00:00:00:00:00:0a
3	0.005389	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=1, hop limit=64 (reply in 4)
4	0.045338	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=1, hop limit=64 (request in 3)
5	1.001622	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=2, hop limit=64 (reply in 6)
6	1.004768	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=2, hop limit=64 (request in 5)
7	2.003304	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=3, hop limit=64 (reply in 8)

Start with what you already know.

1. The NOC team informed you that frames 1 and 2 are legitimate. So, use those as your baseline to detect any abnormalities further down the cap file.
2. Obviously, you are dealing with IPv6 traffic

Study your baseline.

1	0.000000	fe80::2	ff02::1:ff00:1	ICMPv6	86 Neighbor Solicitation for fe80::1 from 00:00:00:00:00:0b
2	0.005346	fe80::1	fe80::2	ICMPv6	86 Neighbor Advertisement fe80::1 (sol, ovr) is at 00:00:00:00:00:0a
3	0.005389	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=1, hop limit=64 (reply in 4)
4	0.045338	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=1, hop limit=64 (request in 3)
5	1.001622	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=2, hop limit=64 (reply in 6)
6	1.004768	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=2, hop limit=64 (request in 5)
7	2.003304	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=3, hop limit=64 (reply in 8)
8	2.004992	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=3, hop limit=64 (request in 7)
9	3.005122	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=4, hop limit=64 (reply in 10)
10	3.008603	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=4, hop limit=64 (request in 9)
11	4.007092	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=5, hop limit=64 (reply in 12)
12	4.009060	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=5, hop limit=64 (request in 11)
13	5.008047	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=6, hop limit=64 (reply in 14)

```
▸ Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
▸ Ethernet II, Src: 00:00:00_00:00:0b (00:00:00:00:00:0b), Dst: IPv6mcast_ff:00:00:01 (33:33:ff:00:00:01)
▸ Internet Protocol Version 6, Src: fe80::2, Dst: ff02::1:ff00:1
▸ Internet Control Message Protocol v6
```

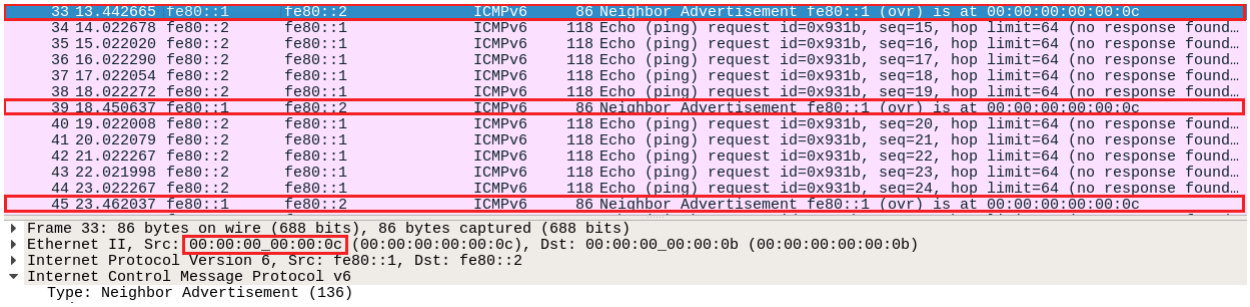
1	0.000000	fe80::2	ff02::1:ff00:1	ICMPv6	86 Neighbor Solicitation for fe80::1 from 00:00:00:00:00:0b
2	0.005346	fe80::1	fe80::2	ICMPv6	86 Neighbor Advertisement fe80::1 (sol, ovr) is at 00:00:00:00:00:0a
3	0.005389	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=1, hop limit=64 (reply in 4)
4	0.045338	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=1, hop limit=64 (request in 3)
5	1.001622	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=2, hop limit=64 (reply in 6)
6	1.004768	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=2, hop limit=64 (request in 5)
7	2.003304	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=3, hop limit=64 (reply in 8)
8	2.004992	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=3, hop limit=64 (request in 7)
9	3.005122	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=4, hop limit=64 (reply in 10)
10	3.008603	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=4, hop limit=64 (request in 9)
11	4.007092	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=5, hop limit=64 (reply in 12)
12	4.009060	fe80::1	fe80::2	ICMPv6	118 Echo (ping) reply id=0x931b, seq=5, hop limit=64 (request in 11)
13	5.008047	fe80::2	fe80::1	ICMPv6	118 Echo (ping) request id=0x931b, seq=6, hop limit=64 (reply in 14)

```
▸ Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
▸ Ethernet II, Src: 00:00:00_00:00:0a (00:00:00:00:00:0a), Dst: 00:00:00_00:00:0b (00:00:00:00:00:0b)
▸ Internet Protocol Version 6, Src: fe80::1, Dst: fe80::2
▸ Internet Control Message Protocol v6
```



The most important thing to note is the **fe80::1's** layer 2 address which is **00:00:00:00:00:0a**.

Now, start browsing frames one by one, until you reach frame #33. You should see the following:



In frame #33 you notice a Neighbor Advertisement, stating that **fe80::1's** layer 2 address is **00:00:00:00:00:0c,** NOT **00:00:00:00:00:0a** as we noted down based on the legitimate frames. Not only that, but the exact same Neighbor Advertisement is repeated in frames #39 and #45. Additionally, this suspicious Neighbor Advertisement seems to be transmitted at frequent 5 sec. intervals.

We can safely conclude we are dealing with a Neighbor Spoofing attack here and an incident should be declared.

