Exploring IPv4 in Malicious Traffic

When it comes to Internet Protocol (IP) addressing, there is a ton of content to cover! Despite this lab having a focus on security, I cannot stress enough the importance of having a fundamental understanding of routing. Especially when you are considering the flow of data, there are a set of IP addresses that are worth memorizing:

Private IP Ranges:

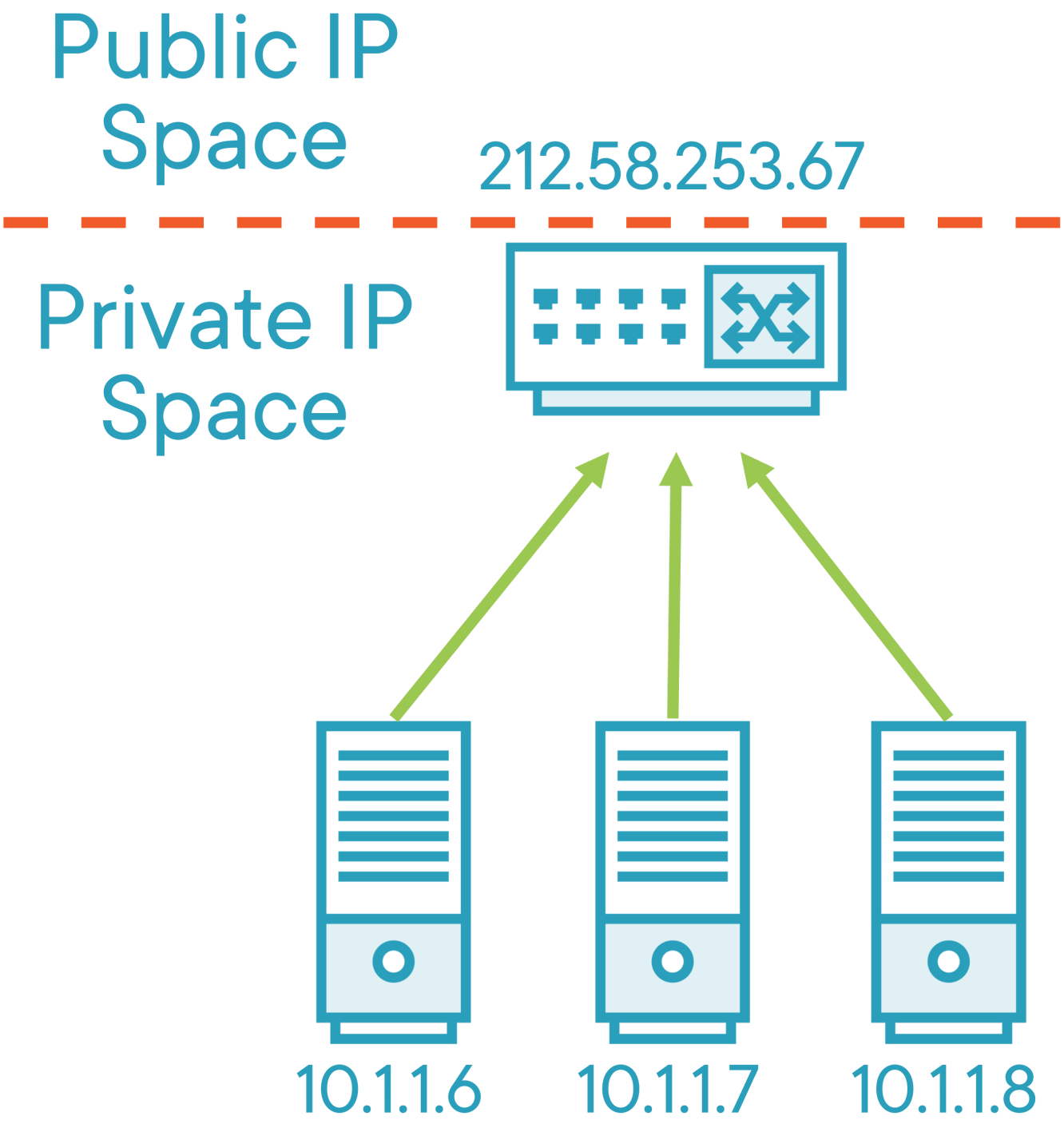
- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Other Important IP Ranges:

- 127.0.0.1 - Localhost Address (Loopback)
- 169.254.0.0/16 - Link-Local Addresses (APIPA)
- 224.0.0.0/4 - IP Multicast

Note: For a complete listing, check out Wikipedia's page on IPv4: https://en.wikipedia.org/wiki/IPv4

It's also important to note that most networks use some form of Network Address Translation (NAT). This usually happens at a routing device where internal private IPs are converted to a single publicly routable IP for Internet communication. Each enterprise will vary in configuration, but the important point is to understand your network's configuration when analyzing the network traffic.



To gain a better understanding of the IPv4 protocol as it relates to security, you will be analyzing a Man-in-the-Middle (MiTM) attack and taking a close look at the IPv4 header fields to identify anomalies.

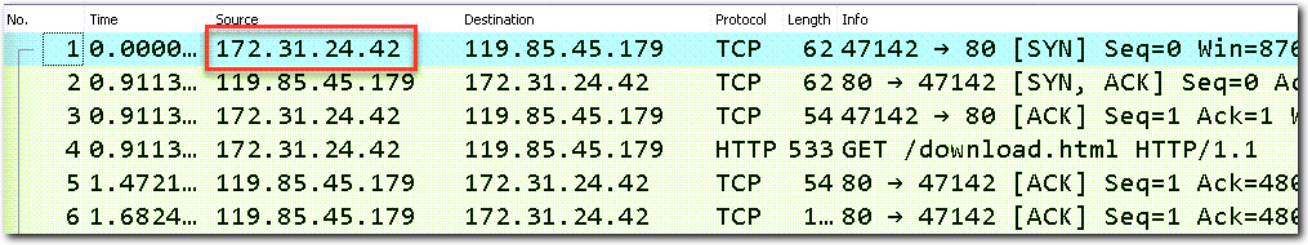
For this next challenge, you will be starting on the Windows 2019 Endpoint.

1. To begin, open Wireshark again if it's not still open from the previous challenge. Now open the **mitm.pcap** file:

`File --> Open`, and double-click: **mitm.pcap**

Note: The file is located on the Desktop in a folder called **LAB_FILES**.

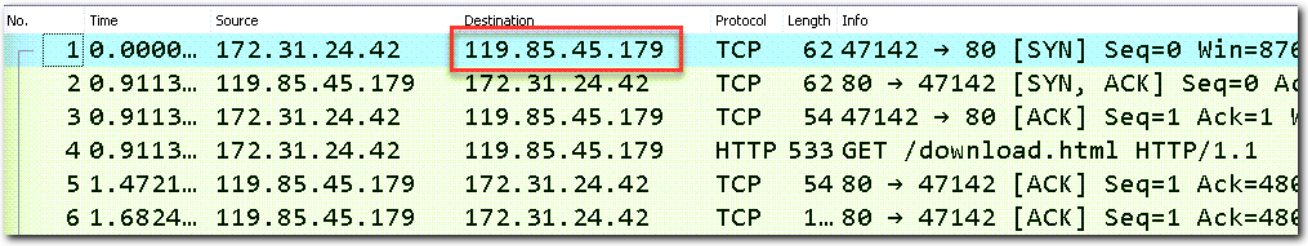
2. For this example, you are analyzing traffic where an internal endpoint is reaching out to pull down a critical resource. What is the internal client's IP address?



No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000…	172.31.24.42	119.85.45.179	TCP	62	47142 → 80 [SYN] Seq=0 Win=876
2	0.9113…	119.85.45.179	172.31.24.42	TCP	62	80 → 47142 [SYN, ACK] Seq=0 Ac
3	0.9113…	172.31.24.42	119.85.45.179	TCP	54	47142 → 80 [ACK] Seq=1 Ack=1 W
4	0.9113…	172.31.24.42	119.85.45.179	HTTP	533	GET /download.html HTTP/1.1
5	1.4721…	119.85.45.179	172.31.24.42	TCP	54	80 → 47142 [ACK] Seq=1 Ack=480
6	1.6824…	119.85.45.179	172.31.24.42	TCP	1…	80 → 47142 [ACK] Seq=1 Ack=480

Analysis: The internal client's IP address is 172.31.24.42. This can be determined by a few factors, but most notably, the fact the IP starts with 172.31 - this let's you know it is in a private IP range. When capturing and analyzing traffic on a network, you will most likely see private IP addressing used for internal clients with some form of NAT at the boundary to communicate externally.

3. The critical resource is hosted on an external IP address. What is that server's IP?

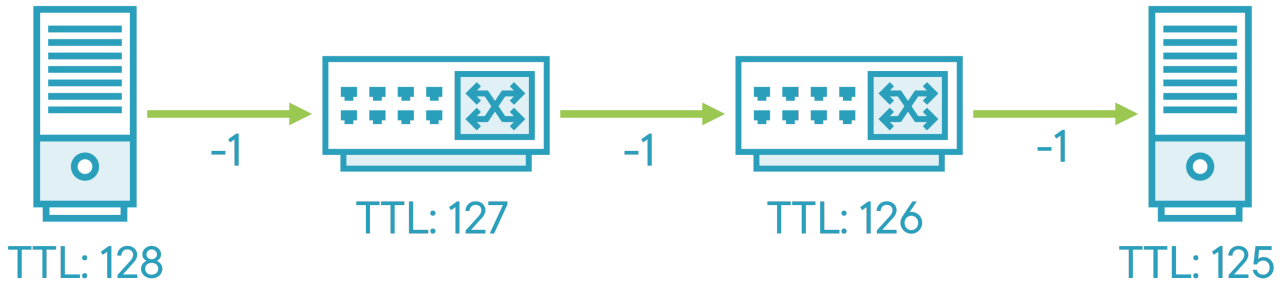


No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000…	172.31.24.42	119.85.45.179	TCP	62	47142 → 80 [SYN] Seq=0 Win=876
2	0.9113…	119.85.45.179	172.31.24.42	TCP	62	80 → 47142 [SYN, ACK] Seq=0 Ac
3	0.9113…	172.31.24.42	119.85.45.179	TCP	54	47142 → 80 [ACK] Seq=1 Ack=1 W
4	0.9113…	172.31.24.42	119.85.45.179	HTTP	533	GET /download.html HTTP/1.1
5	1.4721…	119.85.45.179	172.31.24.42	TCP	54	80 → 47142 [ACK] Seq=1 Ack=480
6	1.6824…	119.85.45.179	172.31.24.42	TCP	1…	80 → 47142 [ACK] Seq=1 Ack=480

Analysis: The external IP address is 119.85.45.179. Further inspection of the traffic and you may realize this is hosting the webpage: http://www.ethereal.com. Though this website hosts something else now, Ethereal was the original name of Wireshark. For the sake of our scenario, let's pretend this external IP address hosts something very critical to our environment.

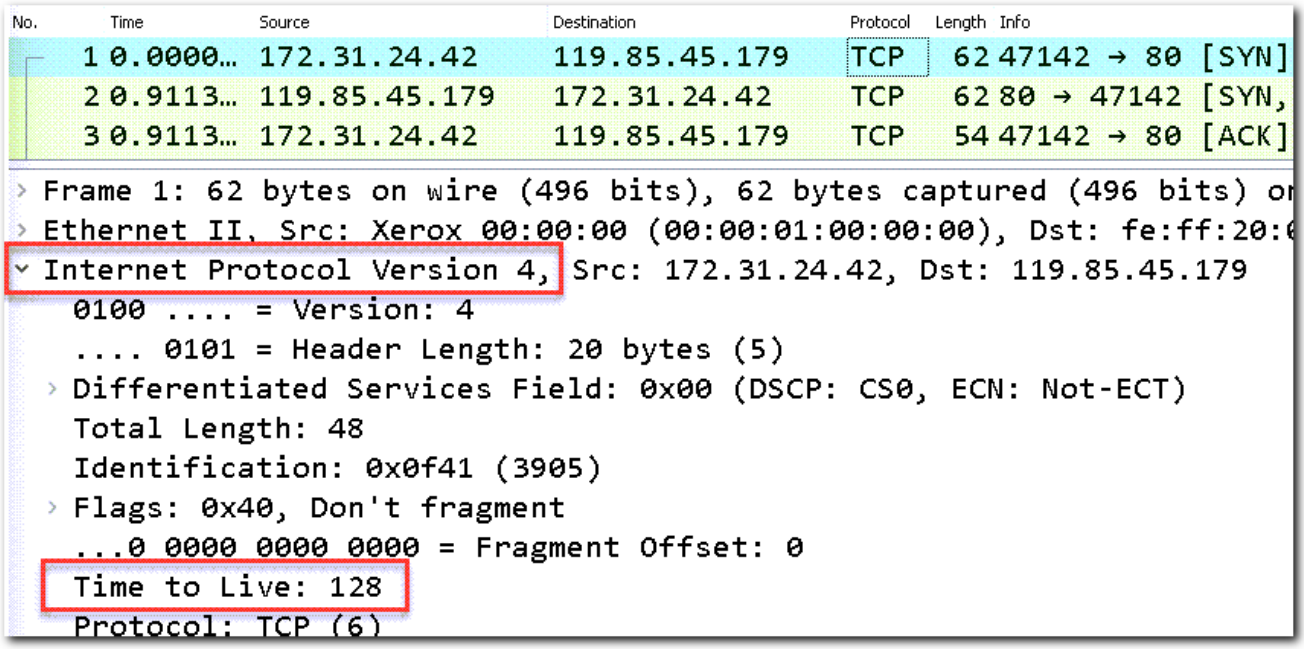
Now that you understand the main two IP addresses involved in the conversations, let's take a closer look at the IPv4 header information. One responsibility of the IPv4 protocol is ensuring cross-network communication is successful. In the event of a routing loop or an instance where a packet can't find it's intended destination, the Time To Live (TTL) field ensures packets can be safely dropped.

Essentially, each packet is given a starting TTL value. As the packet is passed through each endpoint or networking device, the TTL is decremented by one.



4. Back to the network traffic, what is the TTL value of the internal client's packets?

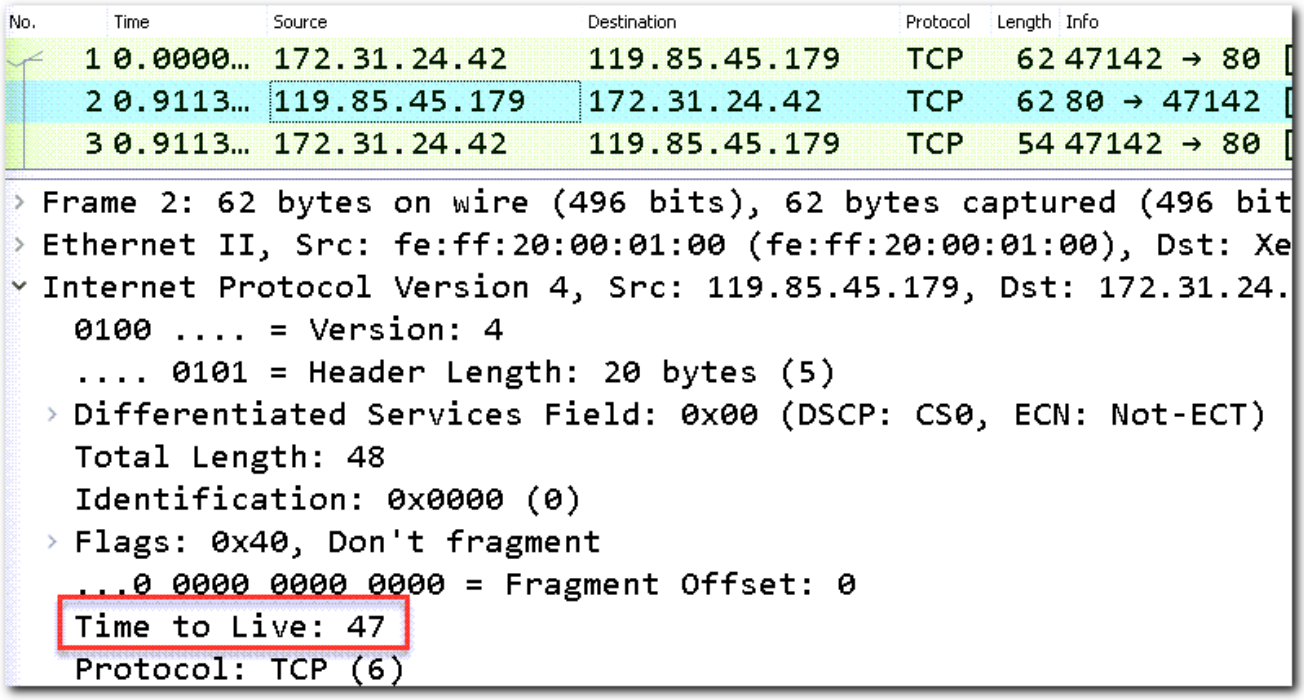
   Note: You can find this by selecting a packet where **172.31.24.42** is in the source column, expanding the IPv4 details section, and looking for "Time to Live".



   Analysis: The TTL value from the internal client is 128. Since this packet capture happened on the client itself, the packets haven't gone anywhere, so you can see the originating TTL values. By default, all Windows operating systems will use 128 as the default TTL value. This is useful to understand when you start analyzing different enumeration techniques.

5. What is the TTL value of the returned server's packets?

   Note: You'll need to select a packet where **119.85.45.179** is in the source column.



Analysis: The returning TTL value is 47. It's important to understand that the server may not be using the same starting TTL value as your client. Each operating system and networking device uses their own defaults, and these values can always be changed.

6. Using the information you just gathered, can you determine how many hops it takes for a packet to traverse between these two hosts? You'll have to make some assumptions, but you can use the default values provided here as a guide:

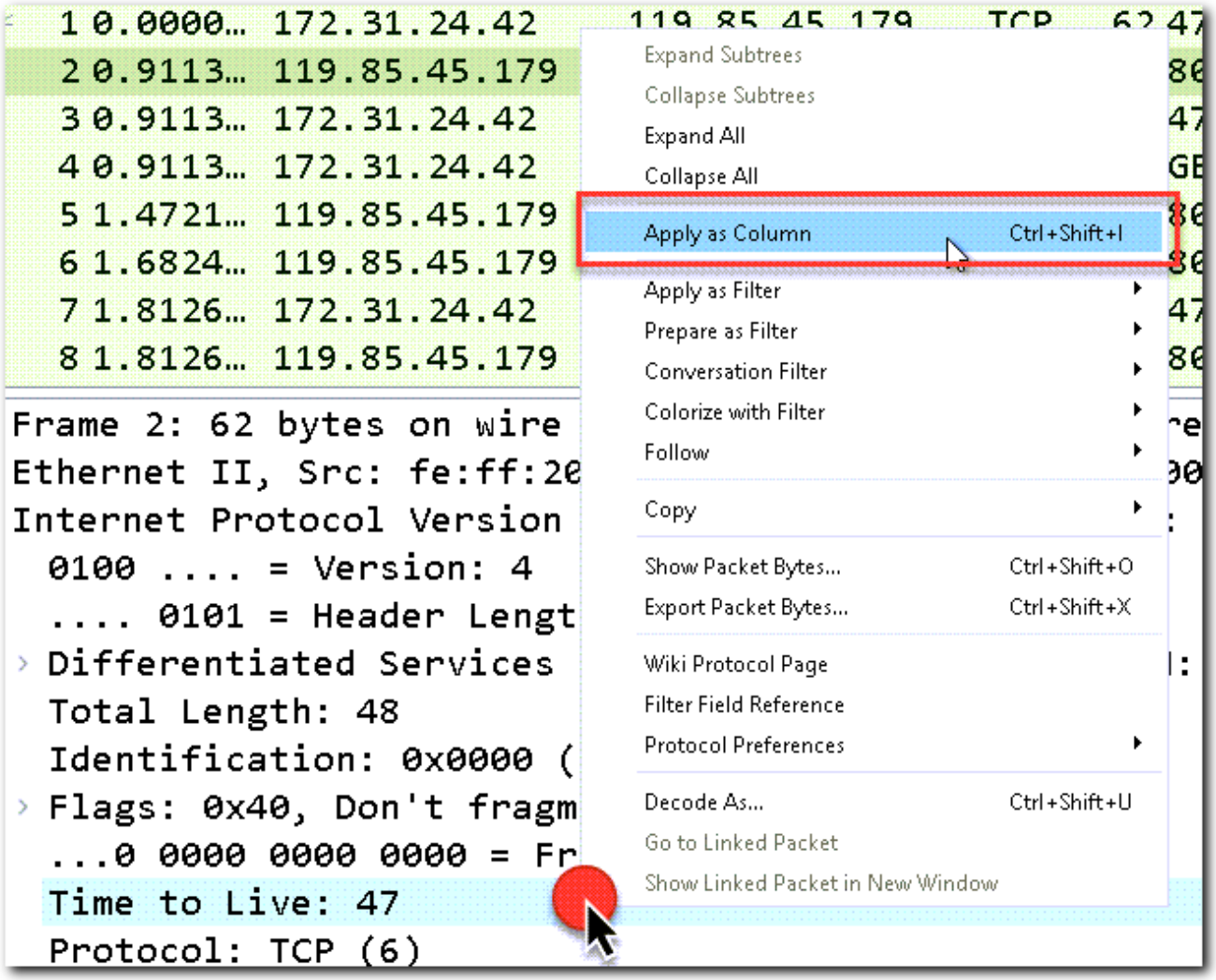
   Note: Default TTL Values:

   - Windows (98 and newer): 128
   - Linux (including most modern *nix operating systems): 64
   - Solaris/AIX: 254

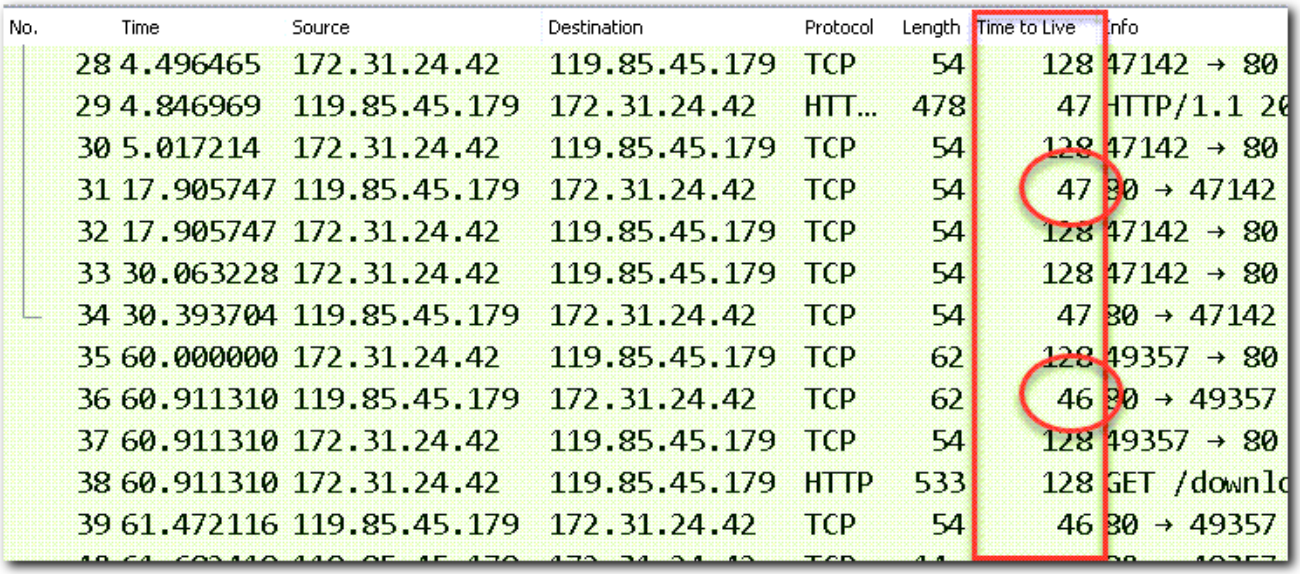
   Analysis: Since most devices use either 64, 128, or 254, you can assume since we are getting a packet with a TTL of 47 returned, that the hosting server probably defaults to 64. (64-47=17) Therefore, this packet traverses through 17 devices to reach the intended destination. Most security analysts won't be performing this calculation for every connection, but it is important to understand the concept and baseline the number of hops for critical devices or services.

7. In order to detect any potential anomalies with this field, add the TTL value as an extra column in Wireshark by right-clicking the Time to Live field and selecting Apply as Column

   Note: This method can be used for any field in Wireshark to assist with analysis.



8. Using the TTL column as a guide, scroll down through the packets to see if you can find any anomalies.



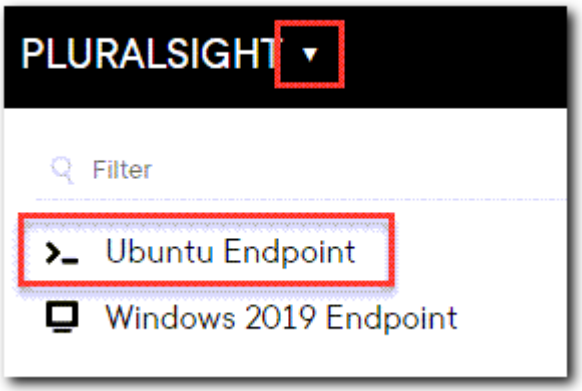
No.	Time	Source	Destination	Protocol	Length	Time to Live	Info
28	4.496465	172.31.24.42	119.85.45.179	TCP	54	128	47142 → 80
29	4.846969	119.85.45.179	172.31.24.42	HTT…	478	47	HTTP/1.1 20
30	5.017214	172.31.24.42	119.85.45.179	TCP	54	128	47142 → 80
31	17.905747	119.85.45.179	172.31.24.42	TCP	54	47	80 → 47142
32	17.905747	172.31.24.42	119.85.45.179	TCP	54	128	47142 → 80
33	30.063228	172.31.24.42	119.85.45.179	TCP	54	128	47142 → 80
34	30.393704	119.85.45.179	172.31.24.42	TCP	54	47	80 → 47142
35	60.000000	172.31.24.42	119.85.45.179	TCP	62	128	49357 → 80
36	60.911310	119.85.45.179	172.31.24.42	TCP	62	46	80 → 49357
37	60.911310	172.31.24.42	119.85.45.179	TCP	54	128	49357 → 80
38	60.911310	172.31.24.42	119.85.45.179	HTTP	533	128	GET /downlo
39	61.472116	119.85.45.179	172.31.24.42	TCP	54	46	80 → 49357

Analysis: In the second request you can see the returned TTL value change from 47 to 46, meaning there's now an additional device involved in the discussion. This is one method you could potentially discover a MiTM attack since all other artifacts from the second request are the same as the first. It's worth noting that routing changes, network configuration changes, or added latency could also affect the TTL. It is important to understand your network configuration and use the TTL as an indicator where possible.

9. Let's take a look at another way you can analyze this traffic using tcpdump. Start by swapping to the **Ubuntu Endpoint**.

   To switch:

   - `CTRL`+ `ALT`+ `SHIFT` (Windows)
   - `CTRL` + `CMD`+ `SHIFT` (macOS)



10. Now use tcpdump to analyze the TTL values coming from the critical server:

```
tcpdump -nn -r ~/mitm.pcap 'src host 119.85.45.179' -v | grep "ttl"
```

Note: grep is a Linux tool that returns only lines that match a specific string, in this case "ttl"

Note: tcpdump options:

- `-v` adds verbosity, giving you more detailed information about the packets.
- tcpdump allows for filters using the Berkeley Packet Filter syntax. Filters are encapsulated in single quotation marks: `'src host 119.85.45.179'`

```
reading from file mitm.pcap, link-type EN10MB (Ethernet
17:31:22.222534 IP (tos 0x0, ttl 47, id 0, offset 0, fl
17:31:22.783340 IP (tos 0x0, ttl 47, id 49310, offset 0
17:31:22.993643 IP (tos 0x0, ttl 47, id 49311, offset 0
17:31:23.123830 IP (tos 0x0, ttl 47, id 49312, offset 0
17:31:23.754737 IP (tos 0x0, ttl 47, id 49313, offset 0
17:31:23.864896 IP (tos 0x0, ttl 47, id 49314, offset 0
17:31:23.945011 IP (tos 0x0, ttl 47, id 49315, offset 0
17:31:24.205385 IP (tos 0x0, ttl 47, id 49316, offset 0
17:31:24.686076 IP (tos 0x0, ttl 47, id 49317, offset 0
17:31:24.806249 IP (tos 0x0, ttl 47, id 49318, offset 0
17:31:24.946451 IP (tos 0x0, ttl 47, id 49319, offset 0
17:31:25.417128 IP (tos 0x0, ttl 47, id 49320, offset 0
17:31:25.537300 IP (tos 0x0, ttl 47, id 49321, offset 0
17:31:25.667488 IP (tos 0x0, ttl 47, id 49322, offset 0
17:31:25.807689 IP (tos 0x0, ttl 47, id 49323, offset 0
17:31:26.158193 IP (tos 0x0, ttl 47, id 49324, offset 0
17:31:39.216971 IP (tos 0x0, ttl 47, id 49325, offset 0
17:31:51.704928 IP (tos 0x0, ttl 47, id 0, offset 0, fl
17:32:22.222534 IP (tos 0x0, ttl 46, id 0, offset 0, fl
17:32:22.783340 IP (tos 0x0, ttl 46, id 49310, offset 0
17:32:22.993643 IP (tos 0x0, ttl 46, id 49311, offset 0
17:32:23.123830 IP (tos 0x0, ttl 46, id 49312, offset 0
17:32:23.754737 IP (tos 0x0, ttl 46, id 49313, offset 0
17:32:23.864896 IP (tos 0x0, ttl 46, id 49314, offset 0
```

Analysis: Though this is a bit more complicated to see and compare the TTL values, it's useful to understand how to see this information if Wireshark or another analysis tool is unavailable. Optimally, as a security analyst, you would be analyzing network traffic and feeding that data into a centralized location. Visualizations can then be created to make this type of analysis much easier.

Congratulations! You have successfully analyzed some network traffic and hopefully have a better understanding of the IPv4 and ARP protocols!