

Course: Cloud and Network Security  
Name: Neville Ngothe Iregi  
Student No.: CS-CNS10-25054  
Date: Tuesday, 16 September 2025

## Week 1: Using Wireshark to Examine Network Traffic

---

---

## **Introduction**

Wireshark is a software protocol analyzer or “packet sniffer” application, used for network troubleshooting, analysis, software and protocol development, and education. A network sniffer—sometimes referred to as a packet sniffer, packet analyzer, protocol analyzer, or network traffic analyzer—can intercept and analyze network traffic that traverses a digital network. As data streams travel back and forth over the network, the sniffer captures each protocol data unit(PDU) and can decode and analyze its content according to the appropriate RFC or other specifications. Thus, Wireshark is a useful tool for anyone working with networks; it can be invaluable to a network or security engineer, forensic investigator, or even a hacker.

In this lab, I used Wireshark to capture ICMP(Internet Control Message Protocol) data packet IP addresses and Ethernet frame MAC addresses. ICMP operates at the network layer as a supporting protocol to IP (Internet Protocol). ICMP is used for reporting errors and management queries since IP lacks an error-reporting or error-correcting mechanism. ICMP also uses **traceroute** and **ping** to perform network diagnosis. **Ping** issues a series of ICMP echo request packets to the destination to test network connectivity and measure latency.

## **Objectives**

1. Capture and analyze local ICMP data in Wireshark
2. Capture and analyze remote ICMP data in Wireshark

## **Part 1: Capture and Analyze Local ICMP Data in Wireshark**

In part 1 of this lab, I pinged my default gateway/router on the LAN and captured ICMP requests and replies in Wireshark. I also looked inside the frames captured for specific information. This analysis helped me to clarify how packet headers are used to transport data to their destination.

### **Step 1: Retrieve my PC interface addresses**

For this lab, I needed to retrieve my PC IP address and its network interface card(NIC) physical address, also called the MAC address.

- a. In a terminal window, I entered **ifconfig** to get the IP address of my PC interface, its description and MAC address and **ip route show default** to get the address of the default gateway.

```
Kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
neville@kali:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inetb fd17:625c:f037:2:a00:27ff:fe95:879b prefixlen 64 scopeid 0x0<global
l0:
    inet6 fd17:625c:f037:2:5678:630c:74e8:67e1 prefixlen 64 scopeid 0x0<global
al0:
    inet6 fe80::a00:27ff:fe95:879b prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:95:87:9b txqueuelen 1000 (Ethernet)
        RX packets 51 bytes 13447 (13.1 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 113 bytes 21072 (20.5 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 8 bytes 480 (480.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 8 bytes 480 (480.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
(neville@kali)-[~]
```

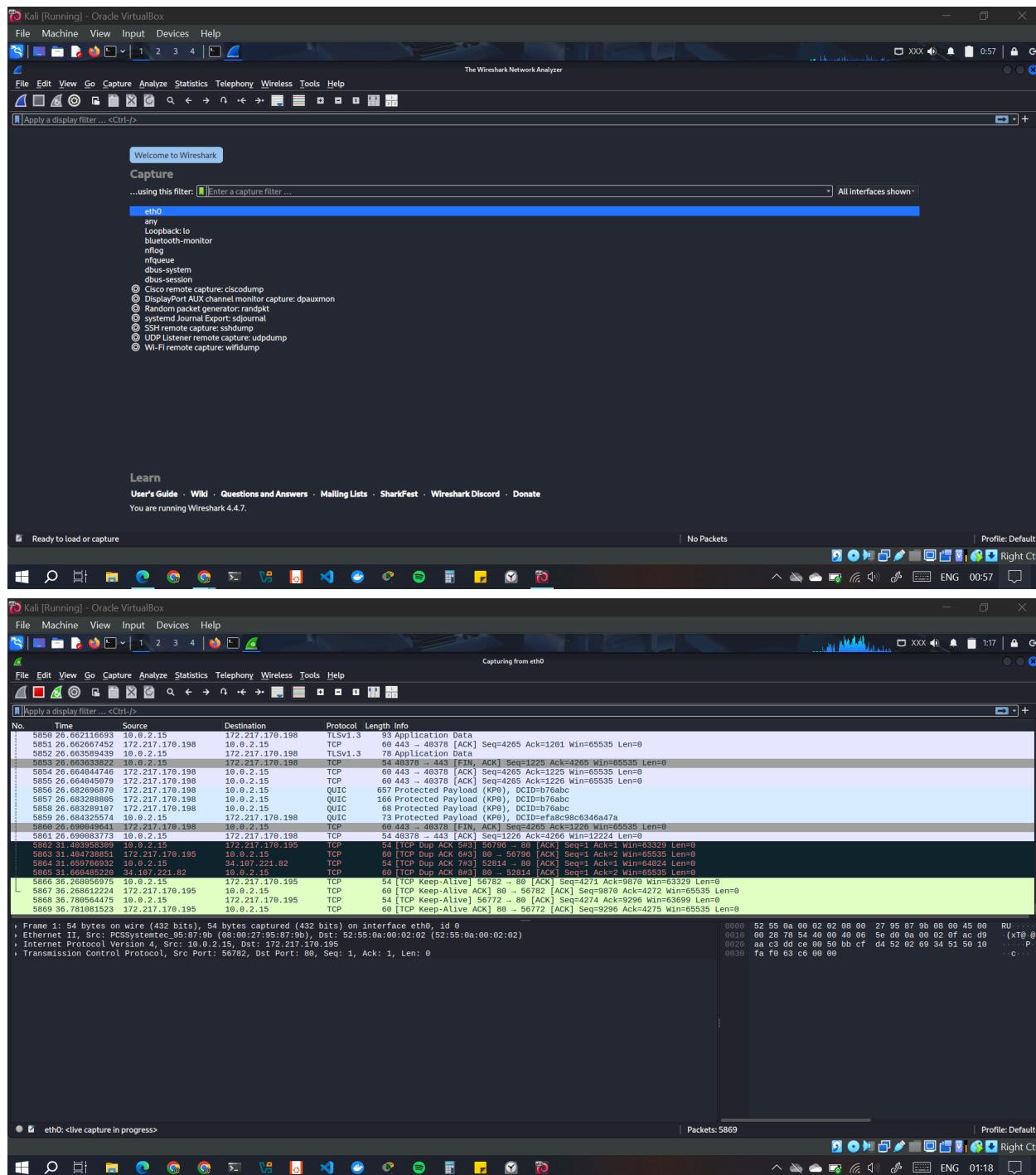
- My IP address is **10.0.2.15** and my MAC address is **08:00:27:95:87:9b**.

```
Kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
neville@kali:~$ ip route show default
default via 10.0.2.2 dev eth0 proto dhcp src 10.0.2.15 metric 100
(neville@kali)-[~]
```

- The address of the default gateway is **10.0.2.2**.

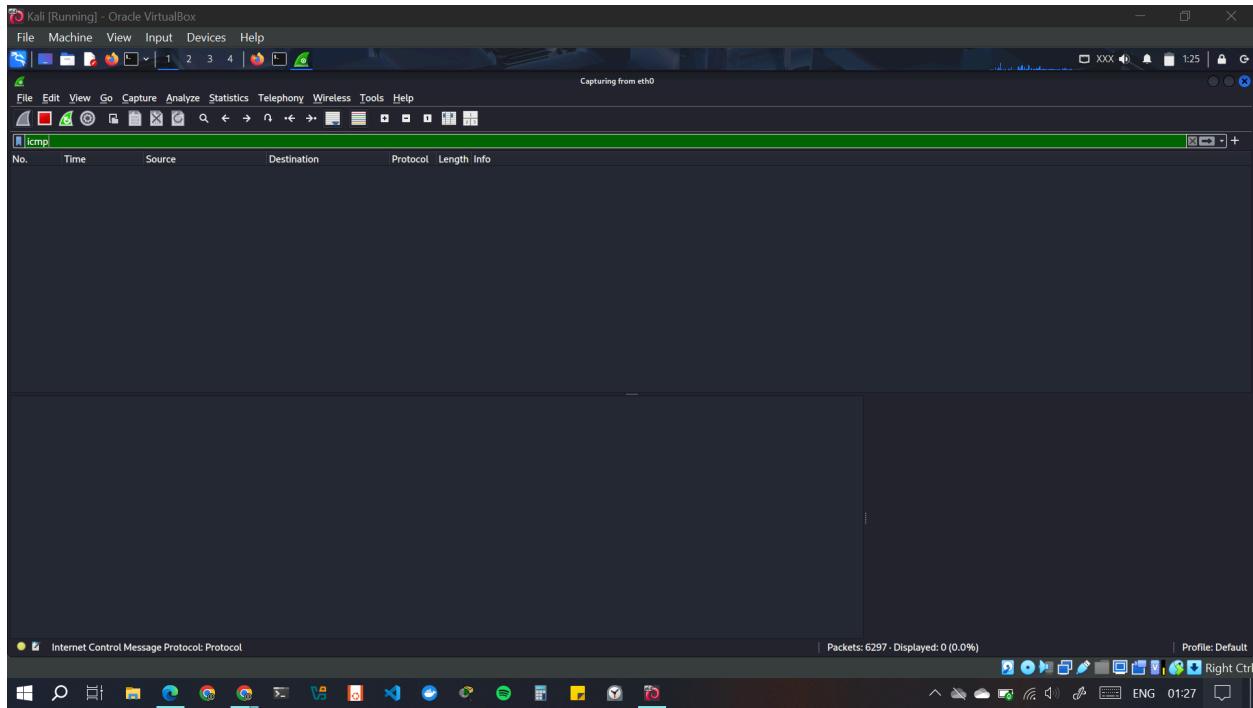
## Step 2: Start Wireshark and begin capturing data.

- I navigated to Wireshark, double-clicked on the desired interface(eth0) to start the packet capture while making sure the desired interface had traffic.
- Information started scrolling down the top section in Wireshark. Data lines appear in different colors based on protocol.

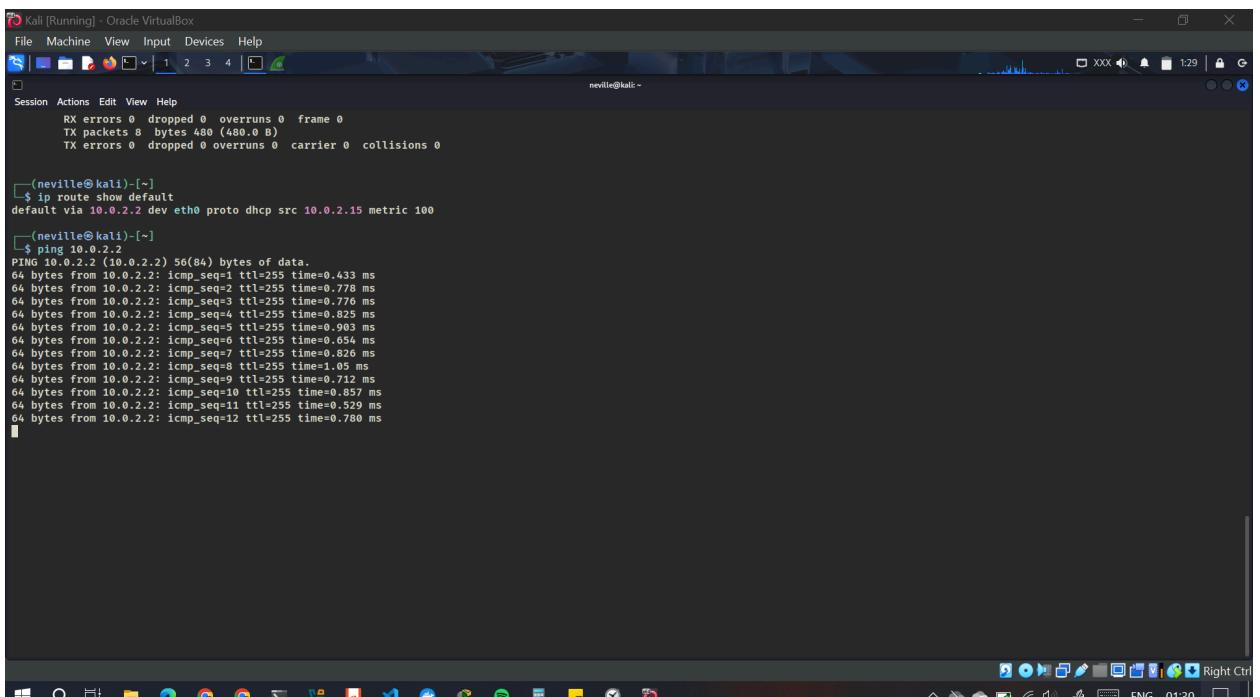


- One can apply a filter to make it easier to view and work with the data being captured by

wireshark. For this lab, we are only interested in displaying ICMP (ping) PDUs. I typed **icmp** in the **Filter** box at the top of Wireshark and pressed Enter to view only ICMP (ping) PDUs.



- c. The icmp filter caused all data in the top window to disappear, but I was still capturing the traffic on the eth0 interface. I navigated to the terminal and pinged the default gateway IP address and at some point, I clicked the **Stop Capture** icon to stop capturing the data.

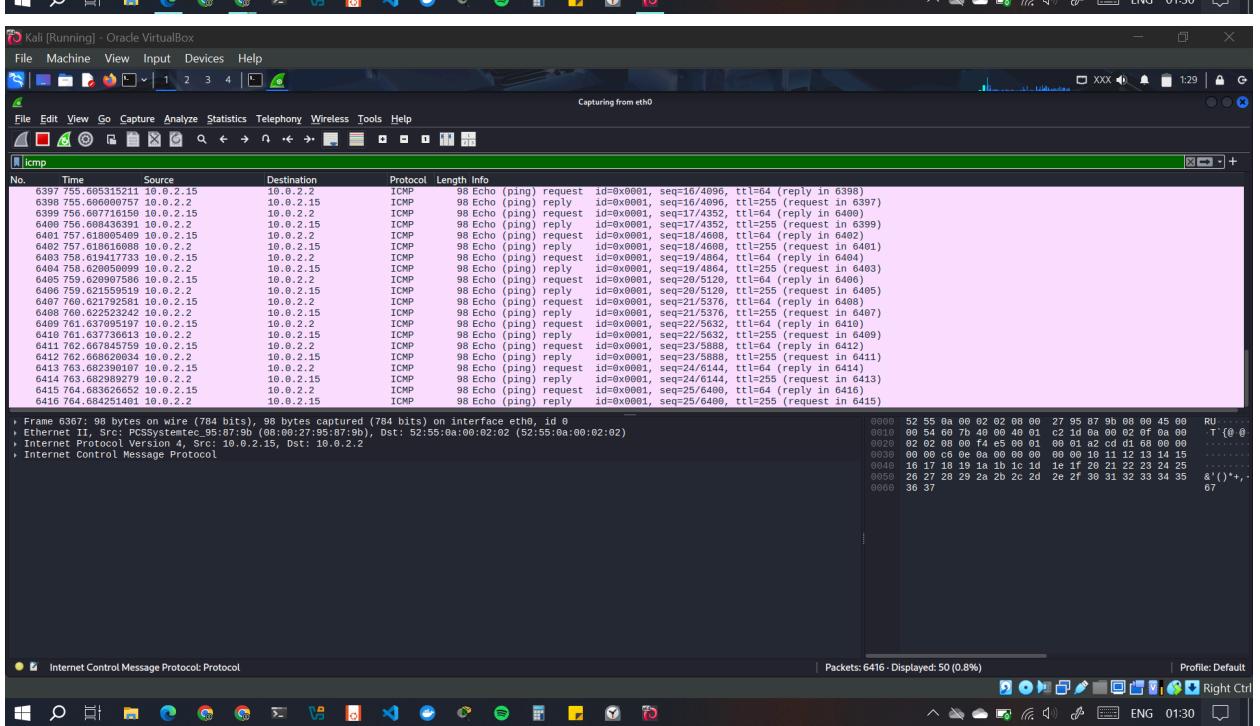


```

(neville@kali)-[~]
$ ip route show default
default via 10.0.2.2 dev eth0 proto dhcp src 10.0.2.15 metric 100

(neville@kali)-[~]
$ ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=255 time=0.433 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=255 time=0.778 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=255 time=0.776 ms
64 bytes from 10.0.2.2: icmp_seq=4 ttl=255 time=0.825 ms
64 bytes from 10.0.2.2: icmp_seq=5 ttl=255 time=0.993 ms
64 bytes from 10.0.2.2: icmp_seq=6 ttl=255 time=0.820 ms
64 bytes from 10.0.2.2: icmp_seq=7 ttl=255 time=0.820 ms
64 bytes from 10.0.2.2: icmp_seq=8 ttl=255 time=1.05 ms
64 bytes from 10.0.2.2: icmp_seq=9 ttl=255 time=0.712 ms
64 bytes from 10.0.2.2: icmp_seq=10 ttl=255 time=0.857 ms
64 bytes from 10.0.2.2: icmp_seq=11 ttl=255 time=0.529 ms
64 bytes from 10.0.2.2: icmp_seq=12 ttl=255 time=0.780 ms

```



Capturing on eth0

No.	Time	Source	Destination	Protocol	Length	Info
6387	755.605315211	10.0.2.15	10.0.2.2	ICMP	88	ping (ping) request id=0x0001, seq=16/4096, ttl=64 (reply in 6398)
6398	755.606009075	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=16/4096, ttl=255 (request in 6397)
6399	756.607716150	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) request id=0x0001, seq=17/4352, ttl=64 (reply in 6400)
6400	756.609436000	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=17/4352, ttl=255 (request in 6399)
6401	757.610153115	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) request id=0x0001, seq=18/4688, ttl=64 (reply in 6402)
6402	757.61018016080	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=18/4688, ttl=255 (request in 6401)
6403	758.619417732	10.0.2.15	10.0.2.2	ICMP	98	Echo (ping) request id=0x0001, seq=19/4864, ttl=64 (reply in 6404)
6404	758.620850699	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=19/4864, ttl=255 (request in 6403)
6405	759.620997586	10.0.2.15	10.0.2.2	ICMP	98	Echo (ping) request id=0x0001, seq=20/5120, ttl=64 (reply in 6406)
6406	760.621723220	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=20/5120, ttl=255 (request in 6405)
6407	760.6217232581	10.0.2.15	10.0.2.2	ICMP	98	Echo (ping) request id=0x0001, seq=21/5376, ttl=64 (reply in 6408)
6408	760.622523242	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=21/5376, ttl=255 (request in 6407)
6409	761.637095197	10.0.2.15	10.0.2.2	ICMP	98	Echo (ping) request id=0x0001, seq=22/5632, ttl=64 (reply in 6410)
6410	761.637095200	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=22/5632, ttl=255 (request in 6409)
6411	762.66862645750	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) request id=0x0001, seq=23/5888, ttl=64 (reply in 6412)
6412	762.6686268034	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=23/5888, ttl=255 (request in 6411)
6413	763.682399107	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) request id=0x0001, seq=24/6144, ttl=64 (reply in 6414)
6414	764.682399107	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=24/6144, ttl=255 (request in 6413)
6415	764.682399107	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) request id=0x0001, seq=25/6400, ttl=64 (reply in 6416)
6416	764.684251401	10.0.2.2	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=25/6400, ttl=255 (request in 6415)

> Frame 6367: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0  
> Ethernet II, Src: PCSystem (00:0c:29:95:87:9b) (00:0c:29:95:87:9b), Dst: 10.0.2.15 (00:0c:29:95:87:9b),  
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.2  
> Internet Control Message Protocol

0000 52 55 08 00 02 02 08 00 27 95 87 9b 08 00 45 00 RU  
0010 00 00 00 00 00 00 00 00 c2 1d 00 00 02 01 00 00 T (0 0  
0020 02 03 00 00 f4 e5 00 00 00 00 00 00 00 00 00 00 00  
0030 00 00 c6 0e 0a 00 00 00 00 00 00 00 00 11 12 33 14 15  
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25  
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35  
0060 36 37 67

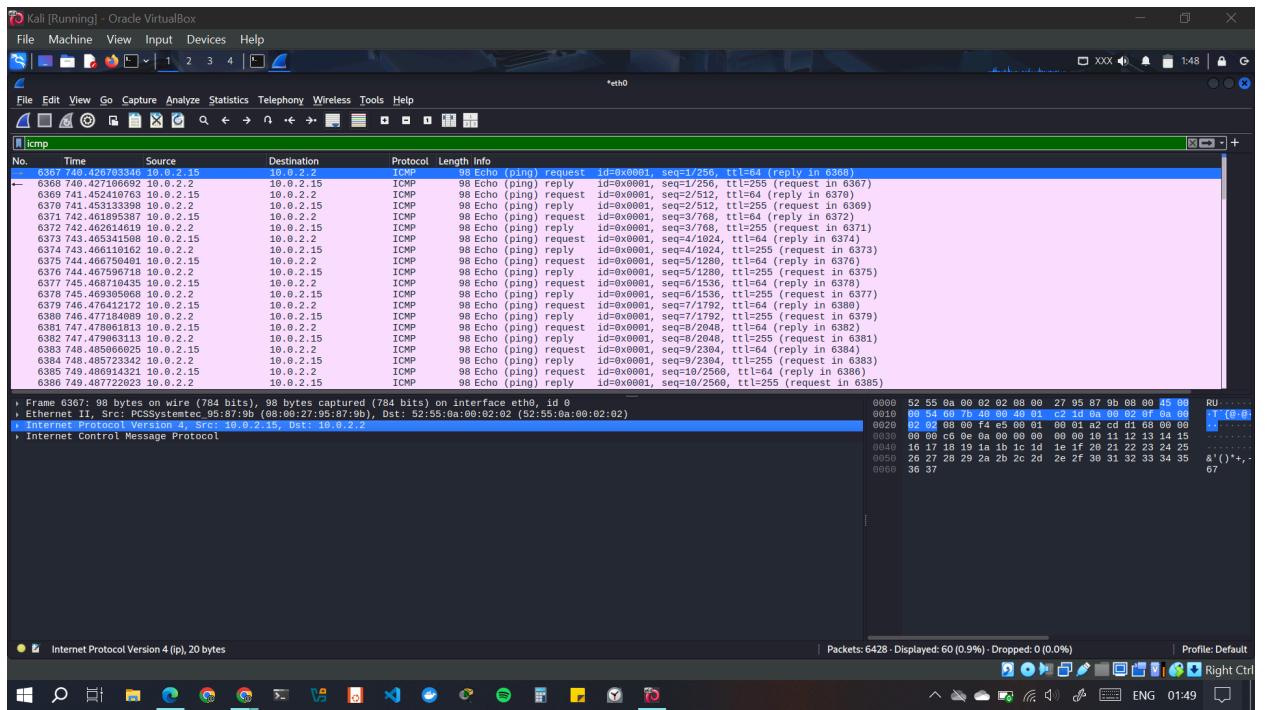
Packets: 6416 - Displayed: 50 (0.8%) | Profile: Default

### Step 3: Examine the captured data

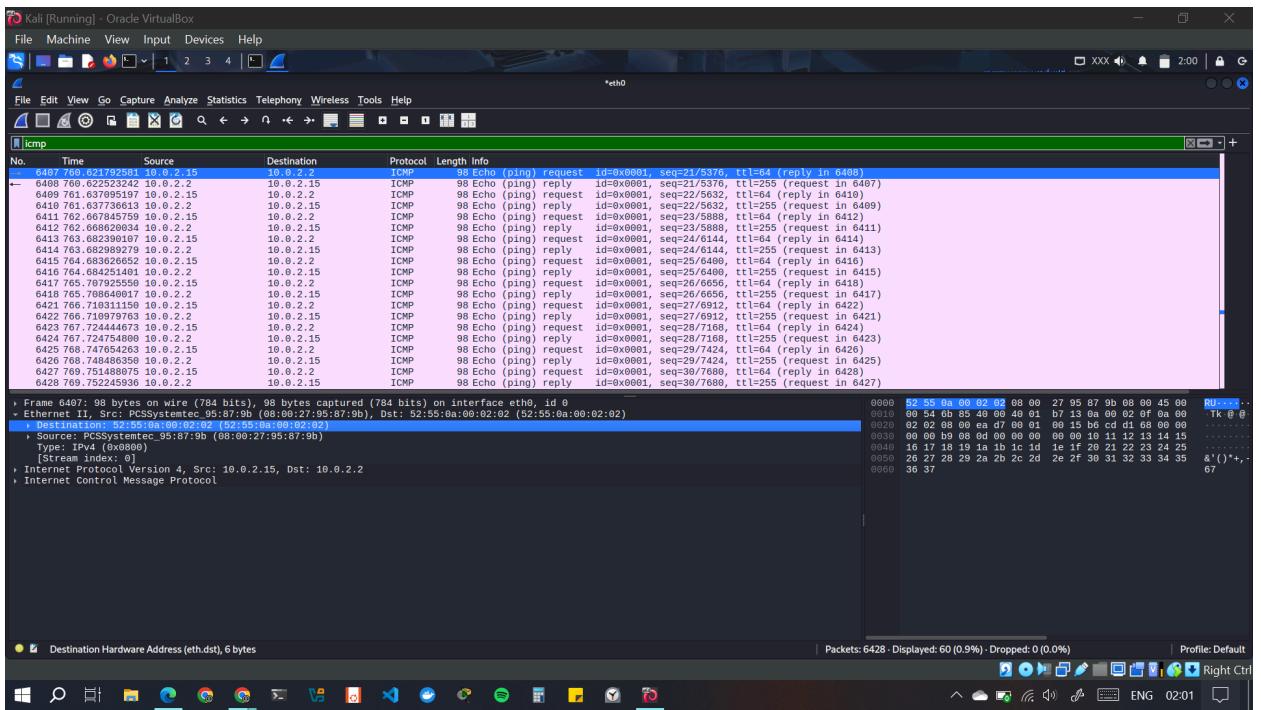
I examined the data that was generated by the ping requests to the default gateway.

Wireshark data is displayed in three sections:

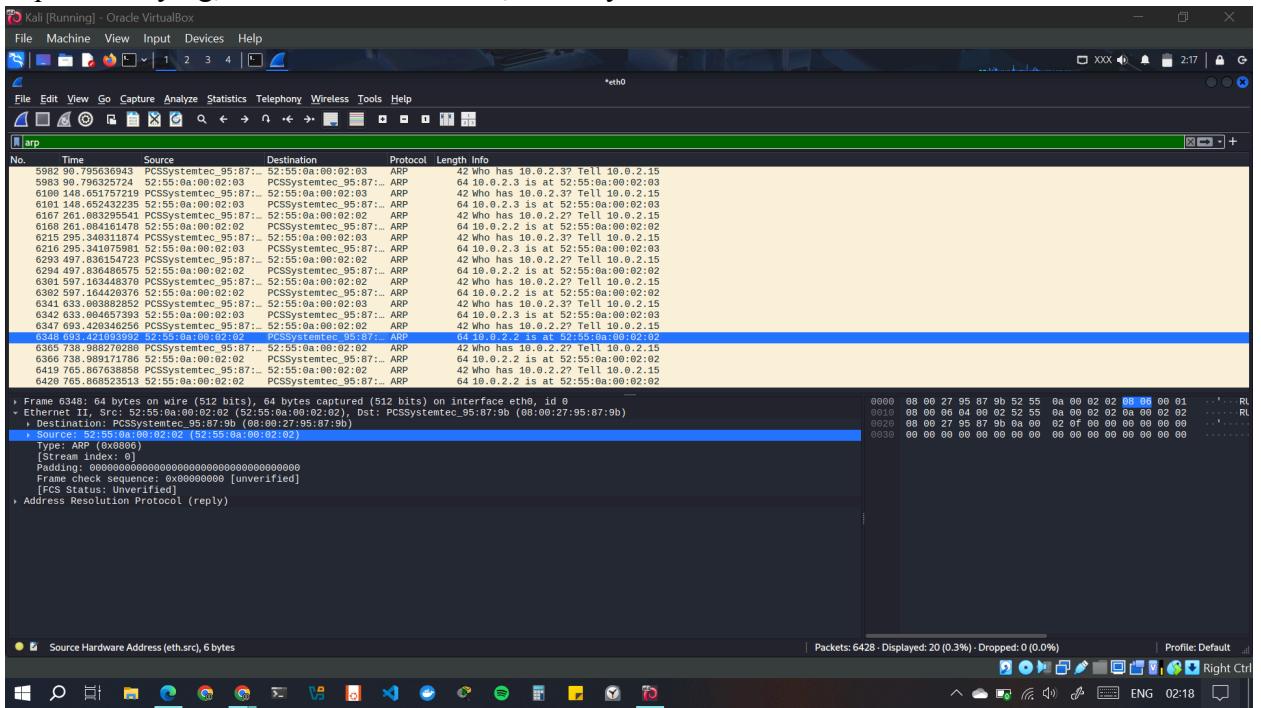
1. The top section displayed the list of PDU frames captured with a summary of the IP packet information listed.
  2. The middle section lists PDU information for the frame selected in the top part of the screen and separates a captured PDU frame by its protocol layers.
  3. The bottom section displays the raw data of each layer. The raw data is displayed in both hexadecimal and decimal form.
- 
- a. I clicked on the first ICMP request PDU frames in the top section of Wireshark. I noticed that the **Source** column has my PC IP address(10.0.2.15) and the **Destination** column contains the IP address of the default gateway that I pinged.

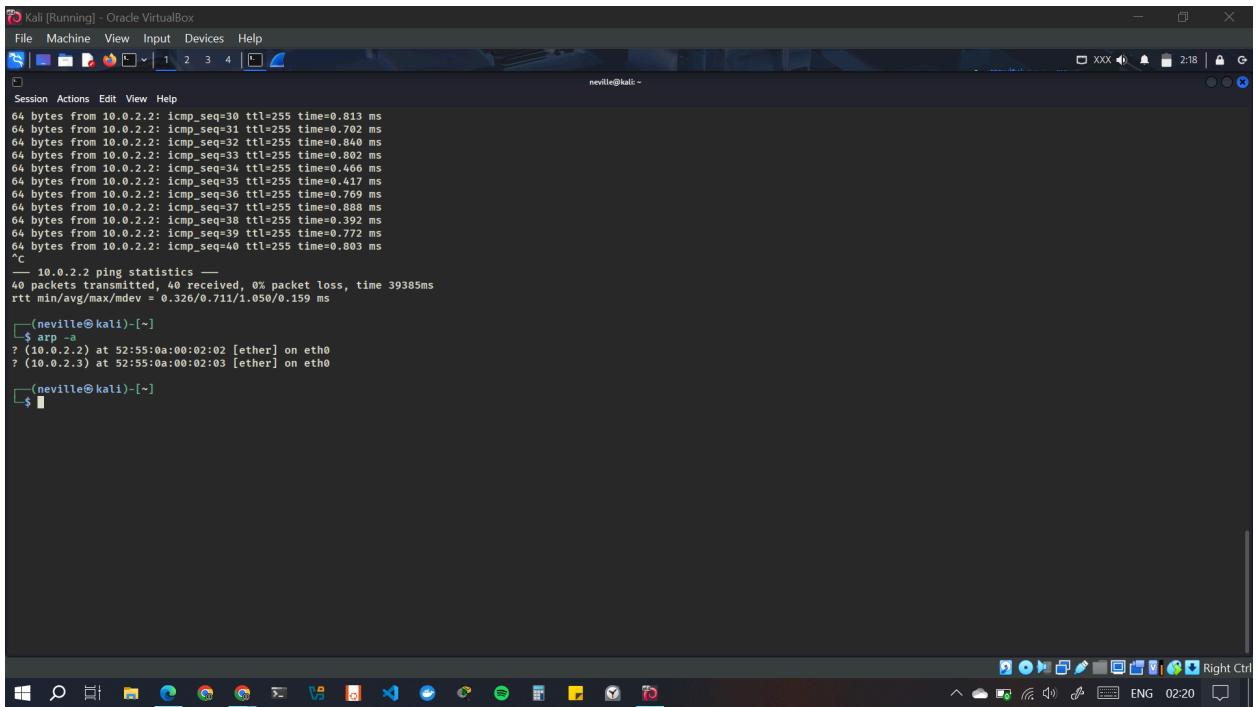


- b. With the PDU frame still selected in the top section, I navigated to the middle section, clicked the arrow to the left of the Ethernet II row to view the destination and source MAC address.



- The source MAC address matched that of my PC interface(**08:00:27:95:87:9b**)
- The MAC address of the destination which is the pinged default gateway is obtained by my PC using an **ARP(Address Resolution Protocol) request** which sends out a broadcast message on LAN to discover the associated MAC address; basically, the ARP request is saying, “Whoever is 10.0.2.2, I need your MAC address”.





Kali [Running] - Oracle VirtualBox

```
File Machine View Input Devices Help
Session Actions Edit View Help
neville@kali:~$ ping 10.0.2.2
64 bytes from 10.0.2.2: icmp_seq=30 ttl=255 time=0.813 ms
64 bytes from 10.0.2.2: icmp_seq=31 ttl=255 time=0.702 ms
64 bytes from 10.0.2.2: icmp_seq=32 ttl=255 time=0.840 ms
64 bytes from 10.0.2.2: icmp_seq=33 ttl=255 time=0.802 ms
64 bytes from 10.0.2.2: icmp_seq=34 ttl=255 time=0.466 ms
64 bytes from 10.0.2.2: icmp_seq=35 ttl=255 time=0.417 ms
64 bytes from 10.0.2.2: icmp_seq=36 ttl=255 time=0.769 ms
64 bytes from 10.0.2.2: icmp_seq=37 ttl=255 time=0.888 ms
64 bytes from 10.0.2.2: icmp_seq=38 ttl=255 time=0.392 ms
64 bytes from 10.0.2.2: icmp_seq=39 ttl=255 time=0.772 ms
64 bytes from 10.0.2.2: icmp_seq=40 ttl=255 time=0.603 ms
'c'
-- 10.0.2.2 ping statistics --
40 packets transmitted, 40 received, 0% packet loss, time 39385ms
rtt min/avg/max/mdev = 0.326/0.711/1.050/0.159 ms

(neville@kali):~$ arp -a
? (10.0.2.2) at 52:55:0a:00:02:02 [ether] on eth0
? (10.0.2.3) at 52:55:0a:00:02:03 [ether] on eth0

(neville@kali):~$
```

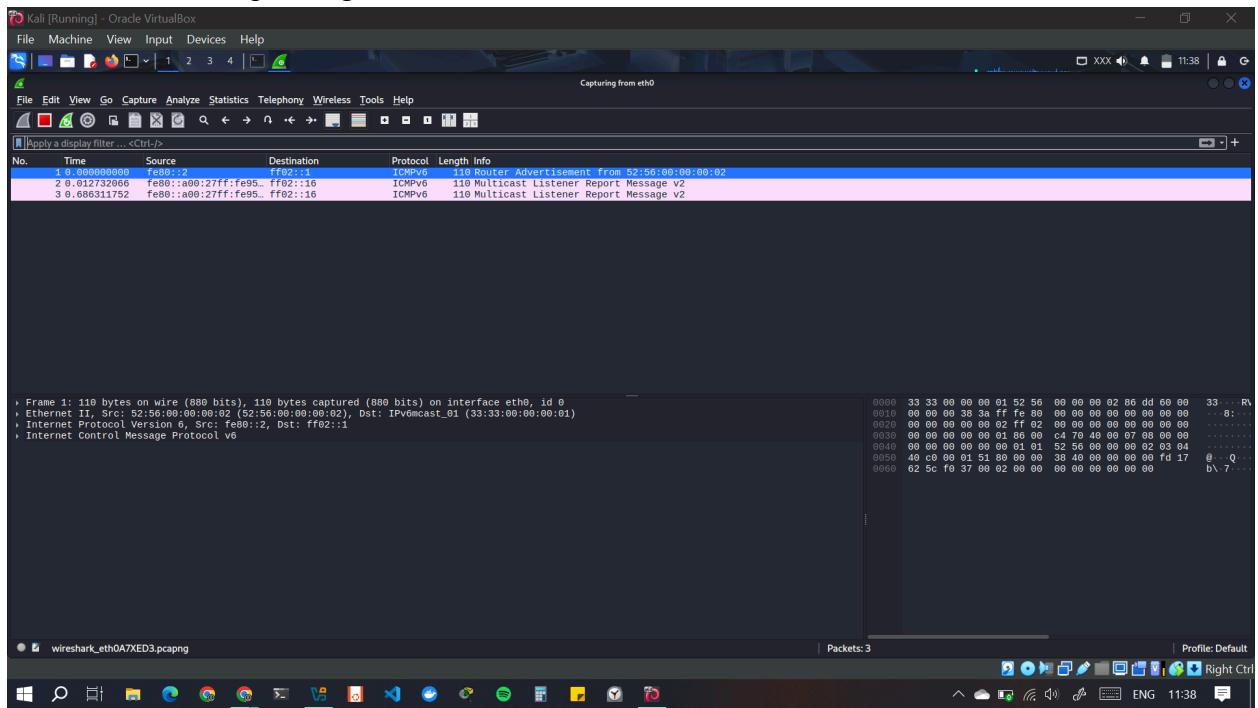
- Using an arp filter in Wireshark, I saw how arp requests for the MAC address of the default gateway. Similarly, using command **arp -a**, I was able to view the ARP table on my system to confirm the default gateway's MAC address, which is **52:55:0a:00:02:02**.

## Part 2: Capture and Analyze Remote ICMP Data In Wireshark

In this part, I pinged remote hosts(hosts not on the LAN) and examined the generated data from those pings. I then determined what is different about this data from the data examined in Part 1.

### **Step 1: Start capturing data on the interface**

- a. I started the data capture again.



- b. With the capture active, I pinged the following 3 website URLs from the terminal.

## 1. [www.yahoo.com](http://www.yahoo.com)

```
Kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
nevilles@kali: ~
Session Actions Edit View Help
ether 08:00:27:95:87:9b txqueuelen 1000 (Ethernet)
RX packets 12 bytes 4029 (3.9 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 38 bytes 5854 (5.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73UP,LOOPBACK,RUNNING mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 12 bytes 680 (680.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 12 bytes 680 (680.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(neville@kali) [~]
$ ping www.yahoo.com
PING me-ycpi-cf-www.g06.yahoodns.net (102.165.180.205) 56(84) bytes of data.
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=1 ttl=255 time=68.9 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=2 ttl=255 time=69.5 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=3 ttl=255 time=69.1 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=4 ttl=255 time=71.4 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=5 ttl=255 time=72.1 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=6 ttl=255 time=69.7 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=7 ttl=255 time=68.7 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=8 ttl=255 time=70.6 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=9 ttl=255 time=68.2 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=10 ttl=255 time=67.7 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=11 ttl=255 time=66.0 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=12 ttl=255 time=83.1 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=13 ttl=255 time=69.2 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=14 ttl=255 time=77.8 ms
64 bytes from e1-ha-ycpi.zaa.yahoo.com (102.165.180.205): icmp_seq=15 ttl=255 time=72.5 ms
^C
--- me-ycpi-cf-www.g06.yahoodns.net ping statistics ---
16 packets transmitted, 15 received, 6.25% packet loss, time 15065ms
rtt min/avg/max/mdev = 66.681/72.140/84.669/5.229 ms
(neville@kali) [~]
$ 
```

2. [www.cisco.com](http://www.cisco.com)

```

Kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
neville@kali: ~
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=7 ttl=255 time=68.7 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=8 ttl=255 time=70.6 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=9 ttl=255 time=68.2 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=10 ttl=255 time=84.7 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=11 ttl=255 time=60.7 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=12 ttl=255 time=83.1 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=13 ttl=255 time=69.2 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=14 ttl=255 time=77.6 ms
64 bytes from ei-ha-ycpl.zaa.yahoo.com (102.165.180.205): icmp_seq=15 ttl=255 time=72.5 ms
`C
-- me-ycpl-cf-www.g08.yahoodns.net ping statistics --
16 packets transmitted, 15 received, 6.25% packet loss, time 15065ms
rtt min/avg/max/mdev = 66.681/72.140/84.669/5.229 ms

(neville@kali): ~]
$ ping www.cisco.com
PING e2807.dsca.akamaiedge.net (2.17.168.94) 56(84) bytes of data.
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=1 ttl=255 time=20.7 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=2 ttl=255 time=23.0 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=3 ttl=255 time=20.2 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=4 ttl=255 time=22.2 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=5 ttl=255 time=22.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=6 ttl=255 time=28.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=7 ttl=255 time=28.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=8 ttl=255 time=22.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=9 ttl=255 time=10 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=10 ttl=255 time=24.1 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=11 ttl=255 time=23.9 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=12 ttl=255 time=22.0 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=13 ttl=255 time=27.4 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=14 ttl=255 time=49.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=15 ttl=255 time=20.4 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=16 ttl=255 time=19.3 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=17 ttl=255 time=21.7 ms
`C
-- e2807.dsca.akamaiedge.net ping statistics --
17 packets transmitted, 17 received, 0% packet loss, time 16036ms
rtt min/avg/max/mdev = 19.289/33.744/109.649/25.604 ms

(neville@kali): ~]
$ 

```

### 3. www.google.com

```

Kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
neville@kali: ~
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=2 ttl=255 time=23.0 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=3 ttl=255 time=23.2 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=4 ttl=255 time=92.2 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=5 ttl=255 time=22.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=6 ttl=255 time=21.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=7 ttl=255 time=28.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=8 ttl=255 time=22.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=9 ttl=255 time=110 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=10 ttl=255 time=24.1 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=11 ttl=255 time=23.9 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=12 ttl=255 time=22.0 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=13 ttl=255 time=27.4 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=14 ttl=255 time=49.8 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=15 ttl=255 time=20.4 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=16 ttl=255 time=19.3 ms
64 bytes from a2-17-168-94.deploy.static.akamaitechnologies.com (2.17.168.94): icmp_seq=17 ttl=255 time=21.7 ms
`C
-- e2807.dsca.akamaiedge.net ping statistics --
17 packets transmitted, 17 received, 0% packet loss, time 16036ms
rtt min/avg/max/mdev = 19.289/33.744/109.649/25.604 ms

(neville@kali): ~]
$ ping www.google.com
PING www.google.com (72.217.170.196) 56(84) bytes of data.
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=1 ttl=255 time=21.1 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=2 ttl=255 time=24.7 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=3 ttl=255 time=23.6 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=4 ttl=255 time=23.3 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=5 ttl=255 time=21.9 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=6 ttl=255 time=19.8 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=7 ttl=255 time=21.6 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=8 ttl=255 time=22.7 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=9 ttl=255 time=23.0 ms
64 bytes from mba01s10-1n-f4.1e100.net (72.217.170.196): icmp_seq=10 ttl=255 time=25.1 ms
`C
-- www.google.com ping statistics --
10 packets transmitted, 10 received, 0% packet loss, time 9026ms
rtt min/avg/max/mdev = 19.810/22.667/25.086/1.534 ms

(neville@kali): ~]
$ 

```

- When I pinged the URLs listed, I noticed that the Domain Name Server(DNS) translates the URL to an IP address.

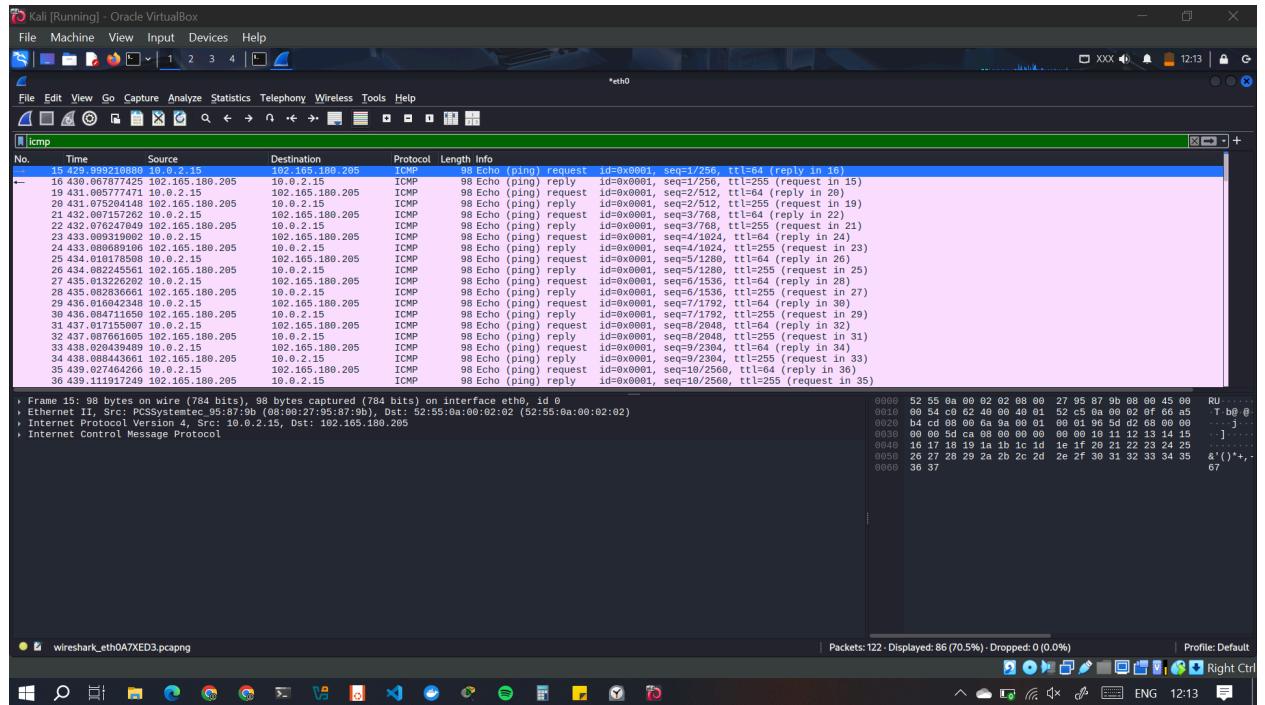
c. In Wireshark, I stopped capturing data by clicking the **Stop Capture** icon.

## Step 2: Examining and analyzing the data from the remote hosts

I reviewed the captured data in Wireshark and examined the IP and MAC addresses of the three locations that I pinged.

- IP address for www.yahoo.com

**102.165.180.205**



- MAC address for www.yahoo.com

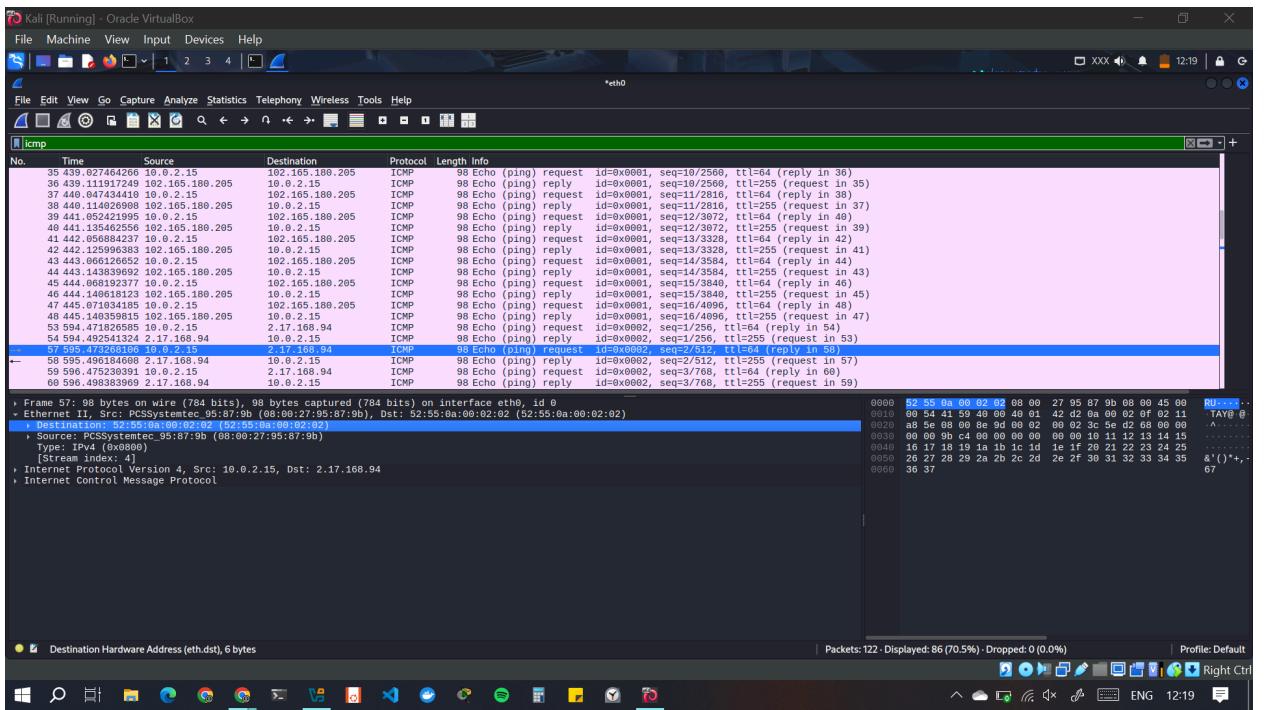
**52:55:0a:00:02:02**

- IP address for www.cisco.com

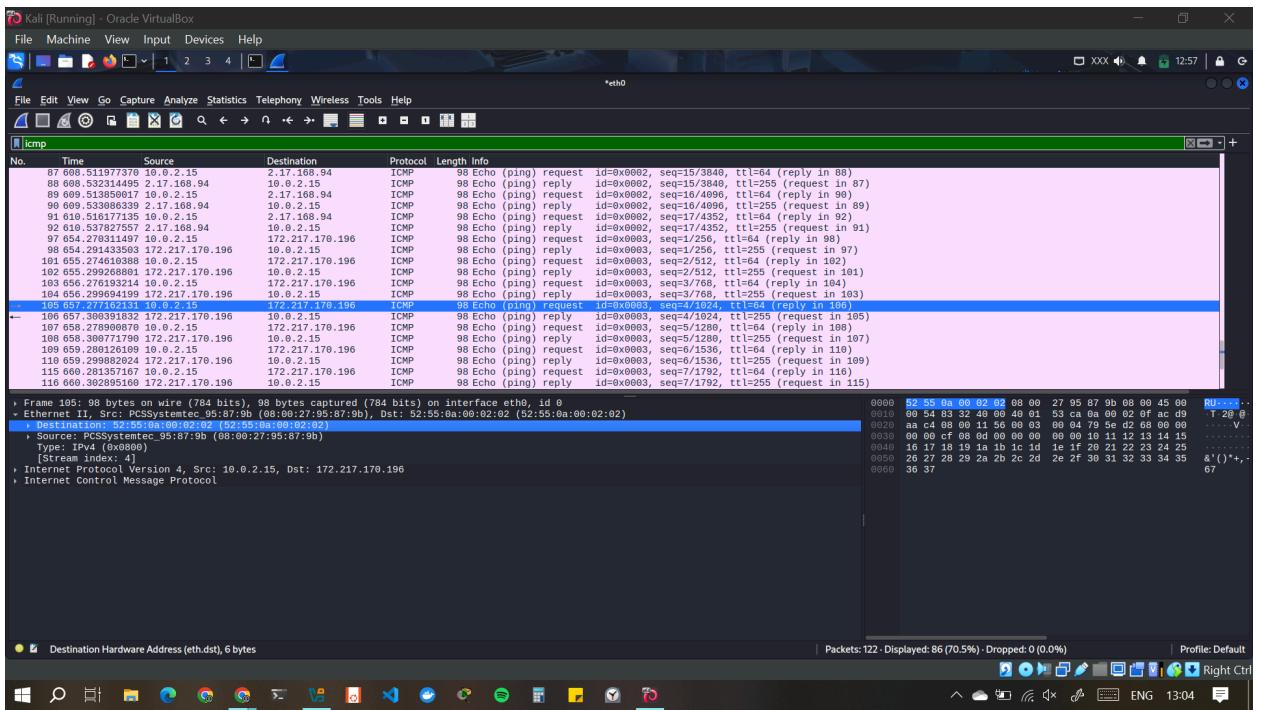
**2.17.168.94**

- MAC address for www.cisco.com

**52:55:0a:00:02:02**



- IP address for [www.google.com](http://www.google.com)  
**172.217.170.196**
- MAC address for [www.google.com](http://www.google.com)  
**52:55:0a:00:02:02**



Question: What is significant about this information?

**Answer: The MAC address is the same for the 3 locations**

Question: How does this information differ from the local ping information you received in Part 1?

**Answer: The information in Part 2 uses one MAC address for the different remote IP addresses meaning that the MAC address belongs to the default gateway through which the remote ping information passes through.**

## REFLECTION QUESTION

Why does Wireshark show the actual MAC address of the local hosts, but not the actual MAC address for the remote hosts?

**Answer:**

**MAC addresses are only meaningful within a local network segment (Layer 2 domain). When you send traffic to a remote host, your packet is first sent to your default gateway (router). That router deencapsulates the Ethernet header (with your MAC + router's MAC) and forwards the packet at Layer 3 (IP). Each hop along the path builds a new**

---

**Ethernet frame with its own source and next-hop destination MAC addresses. By the time your traffic reaches the remote host, your machine has no visibility of its MAC address — only of your local router's MAC.**

**Wireshark shows actual MAC addresses for local hosts because those Ethernet frames are on your LAN. For remote hosts, you only see the MAC of your local gateway/router, since that's the last device your machine directly communicates with at Layer 2. The remote host's actual MAC never leaves its own LAN.**

## **Conclusion**

This lab demonstrated how Wireshark can be used to capture and analyze ICMP traffic at both the local and remote levels. By examining local pings, I confirmed that my system communicates directly with other devices on the LAN using their actual MAC addresses, which are resolved through ARP. However, when pinging remote hosts, Wireshark showed that all ICMP packets shared the same MAC address — that of the default gateway — because MAC addressing is only relevant within a local network segment. Beyond the LAN, packets are forwarded at the IP layer, and new Ethernet frames are created at each hop. This assignment helped to practically show the difference between Layer 2 (MAC) and Layer 3 (IP) addressing, and highlighted the important role of the default gateway in connecting local devices to remote networks.