

OLD DOMINION UNIVERSITY

CYSE 301 CYBERSECURITY TECHNIQUES AND OPERATIONS

Assignment #2: Traffic Tracing and Analysis

Carl Lochstampfor
Student UIN: 01290201
Date: 02/09/2026

Assignment #2: Traffic Tracing and Analysis

Task A: Get started with Wireshark (30 points)

In this task, you will be using Wireshark on External Kali to monitor the traffic when External Kali and Ubuntu VM are talking to each other.

You should keep Wireshark in External / attacker Kali running in the background while performing the following tasks. Connect /Turn on Ubuntu VM (and pfsense VM) as well.

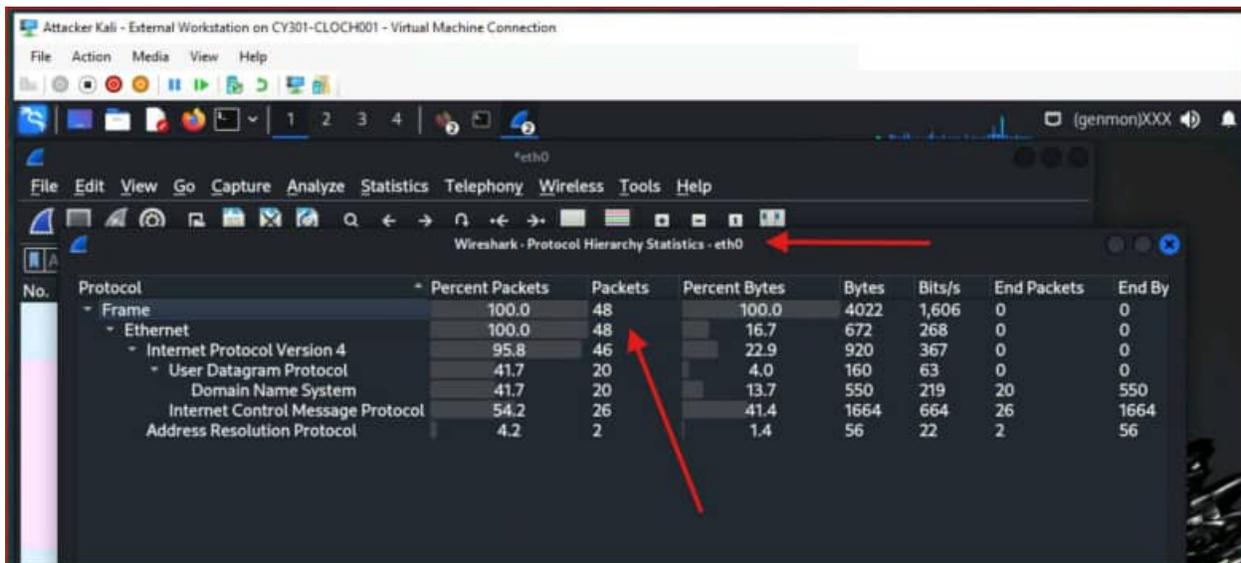
1. Open Wireshark on External / Attacker Kali and listen on interface “eth0”.
2. Open a new **terminal** in Attacker Kali, then **ping the Ubuntu VM for 5 – 10 seconds**.
3. Open a **new web browser tab in Attacker Kali** (even if no webpage will be displayed) and **keep it for a couple of seconds**.
4. **Stop capturing (by clicking the red button on the tool bar)**.

Now, answer the following questions. **You need to provide a screenshot for each questions** that proves the answers to each question you write.

Q1. (5 pts): How many packets are captured in total? How many packets are displayed?

Answer: **48 packets** captured in total. **48 packets** are displayed.

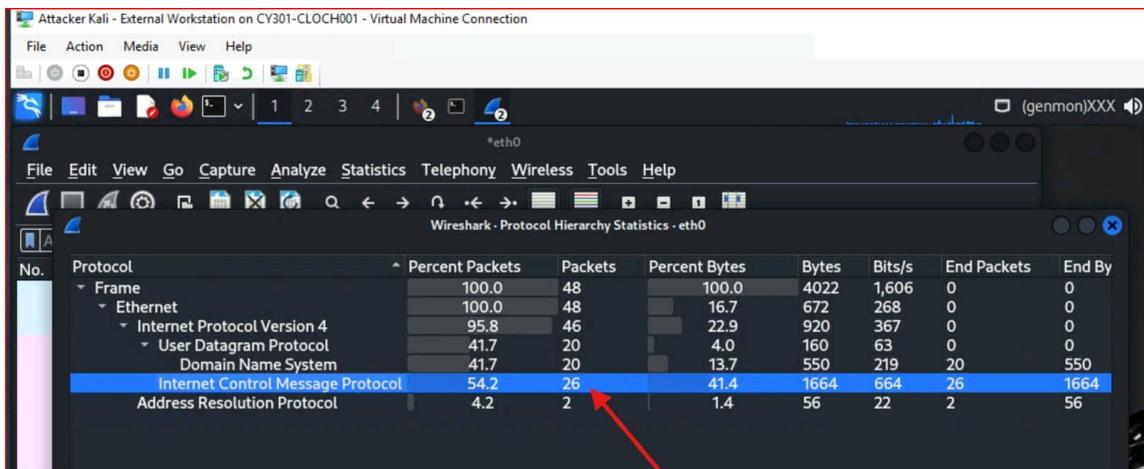
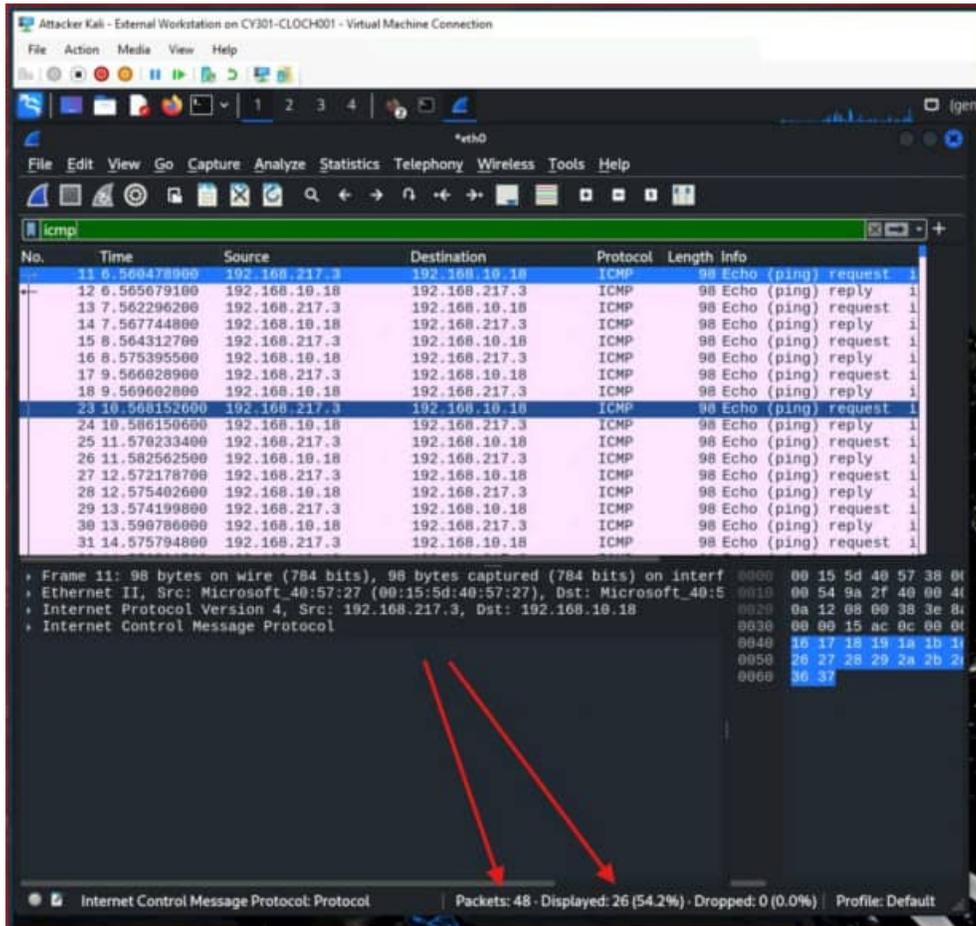
Screenprint:



Q2. (5 pts): Apply “ICMP” as a display filter in Wireshark. Then repeat the previous question (Q1).

Answer: There are 26 ICMP packets out of a 48 total packets captured and displayed.

Screenprint:



Q3. (5 pts): Select an **Echo (reply)** message from the list. What are the **source** and **destination** IPs of this packet? What are the **sequence number** and the **size of the data**? What is the **response time**?

Answer: **Source IP of the packet: 192.168.217.3. Destination IP of the packet: 192.168.10.18. The sequence number is 1 (0x0001) and the size of the data is 40 bytes. The response time is 5.200 ms.**

Screenprint:

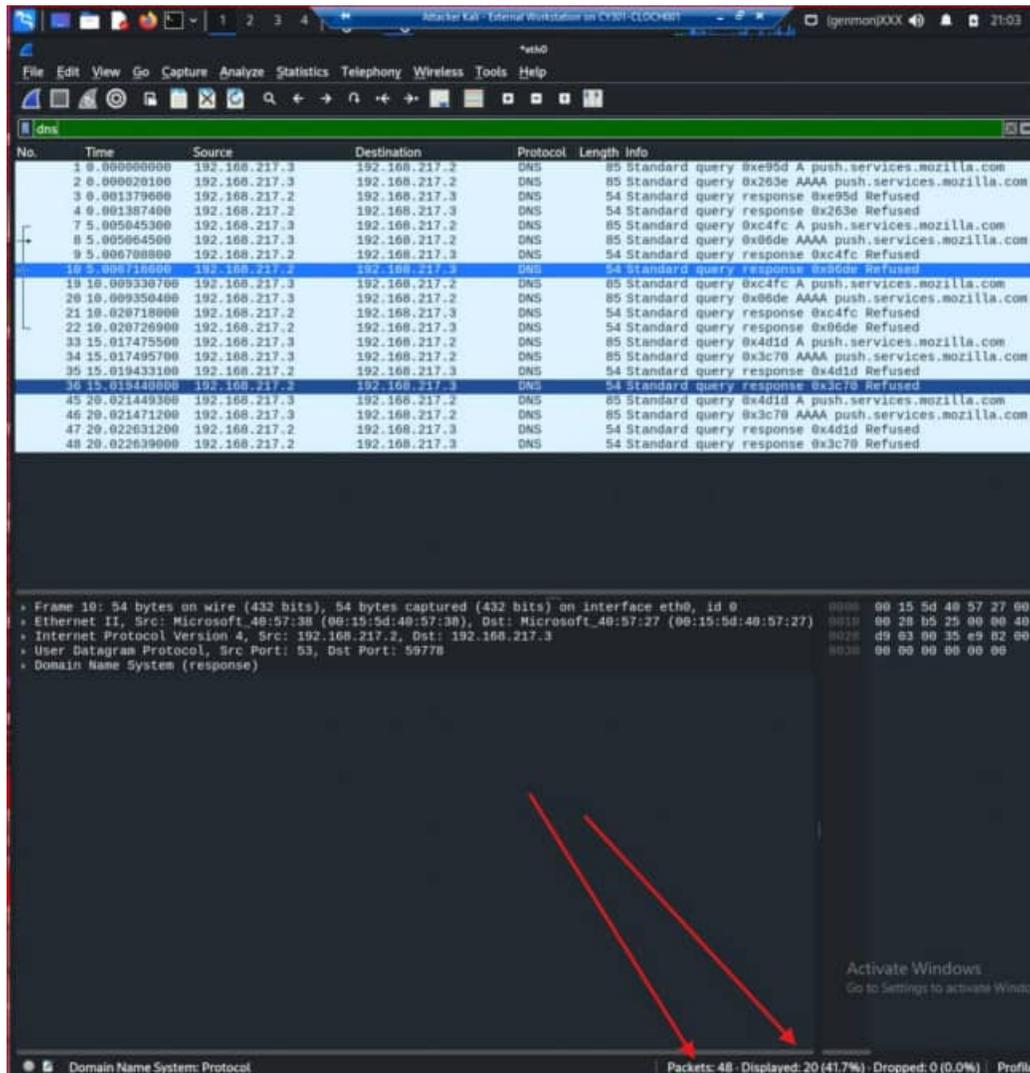
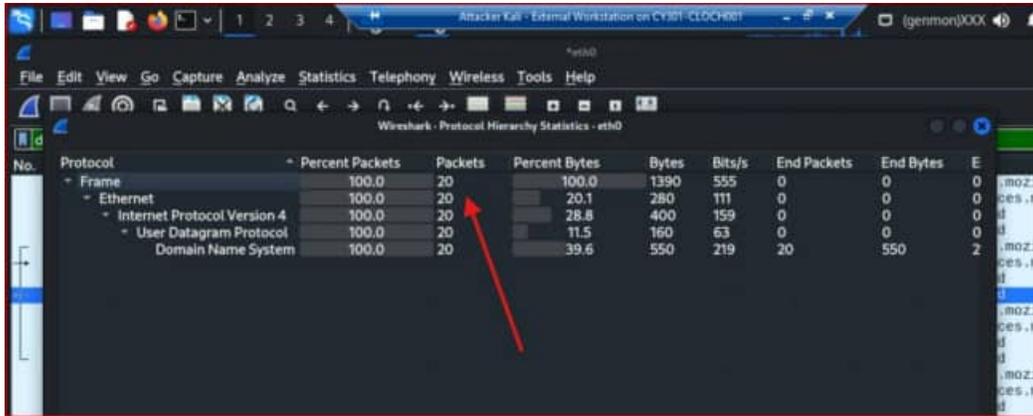
| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|--------------------|--------------------|----------|--------|---|
| 1 | 0.000000000 | 192.168.217.3 | 192.168.217.2 | DNS | 85 | Standard query 0xe95d A push.services.mozilla.com |
| 2 | 0.000020100 | 192.168.217.3 | 192.168.217.2 | DNS | 85 | Standard query 0x263e AAAA push.services.mozilla.com |
| 3 | 0.001379600 | 192.168.217.2 | 192.168.217.3 | DNS | 54 | Standard query response 0xe95d Refused |
| 4 | 0.001387400 | 192.168.217.2 | 192.168.217.3 | DNS | 54 | Standard query response 0x263e Refused |
| 5 | 0.202650700 | Microsoft_40:57:27 | Microsoft_40:57:38 | ARP | 42 | Who has 192.168.217.2? Tell 192.168.217.3 |
| 6 | 0.207523200 | Microsoft_40:57:38 | Microsoft_40:57:27 | ARP | 42 | 192.168.217.2 is at 00:15:5d:40:57:38 |
| 7 | 5.005045300 | 192.168.217.3 | 192.168.217.2 | DNS | 85 | Standard query 0xc4fc A push.services.mozilla.com |
| 8 | 5.005064500 | 192.168.217.3 | 192.168.217.2 | DNS | 85 | Standard query 0x06de AAAA push.services.mozilla.com |
| 9 | 5.006708800 | 192.168.217.2 | 192.168.217.3 | DNS | 54 | Standard query response 0xc4fc Refused |
| 10 | 5.006716600 | 192.168.217.2 | 192.168.217.3 | DNS | 54 | Standard query response 0x06de Refused |
| 11 | 6.560478900 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=1/256, ttl=64 (req) |
| 12 | 6.565679100 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8aed, seq=1/256, ttl=63 (req) |
| 13 | 7.562296200 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=2/512, ttl=64 (req) |
| 14 | 7.567744800 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8aed, seq=2/512, ttl=63 (req) |
| 15 | 8.564312700 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=3/768, ttl=64 (req) |
| 16 | 8.575395500 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8aed, seq=3/768, ttl=63 (req) |
| 17 | 9.566028900 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=4/1024, ttl=64 (req) |
| 18 | 9.569002800 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8aed, seq=4/1024, ttl=63 (req) |
| 19 | 10.009330700 | 192.168.217.3 | 192.168.217.2 | DNS | 85 | Standard query 0xc4fc A push.services.mozilla.com |
| 20 | 10.009350400 | 192.168.217.3 | 192.168.217.2 | DNS | 85 | Standard query 0x06de AAAA push.services.mozilla.com |
| 21 | 10.020718000 | 192.168.217.2 | 192.168.217.3 | DNS | 54 | Standard query response 0xc4fc Refused |
| 22 | 10.020726900 | 192.168.217.2 | 192.168.217.3 | DNS | 54 | Standard query response 0x06de Refused |
| 23 | 10.568152600 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=5/1280, ttl=64 (req) |
| 24 | 10.586150600 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8aed, seq=5/1280, ttl=63 (req) |
| 25 | 11.570233400 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=6/1536, ttl=64 (req) |
| 26 | 11.582562500 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8aed, seq=6/1536, ttl=63 (req) |
| 27 | 12.572178700 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=7/1792, ttl=64 (req) |
| 28 | 12.575402600 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8aed, seq=7/1792, ttl=63 (req) |
| 29 | 13.574199800 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8aed, seq=8/2048, ttl=64 (req) |

Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
 Ethernet II, Src: Microsoft_40:57:38 (00:15:5d:40:57:38), Dst: Microsoft_40:57:27 (00:15:5d:40:57:27)
 Internet Protocol Version 4, Src: 192.168.10.18, Dst: 192.168.217.3
 Internet Control Message Protocol

Q4. (5 pts): Apply “DNS” as a display filter in Wireshark. How many packets are displayed?

Answer: There are **20 packets displayed** out of 48 total packets.

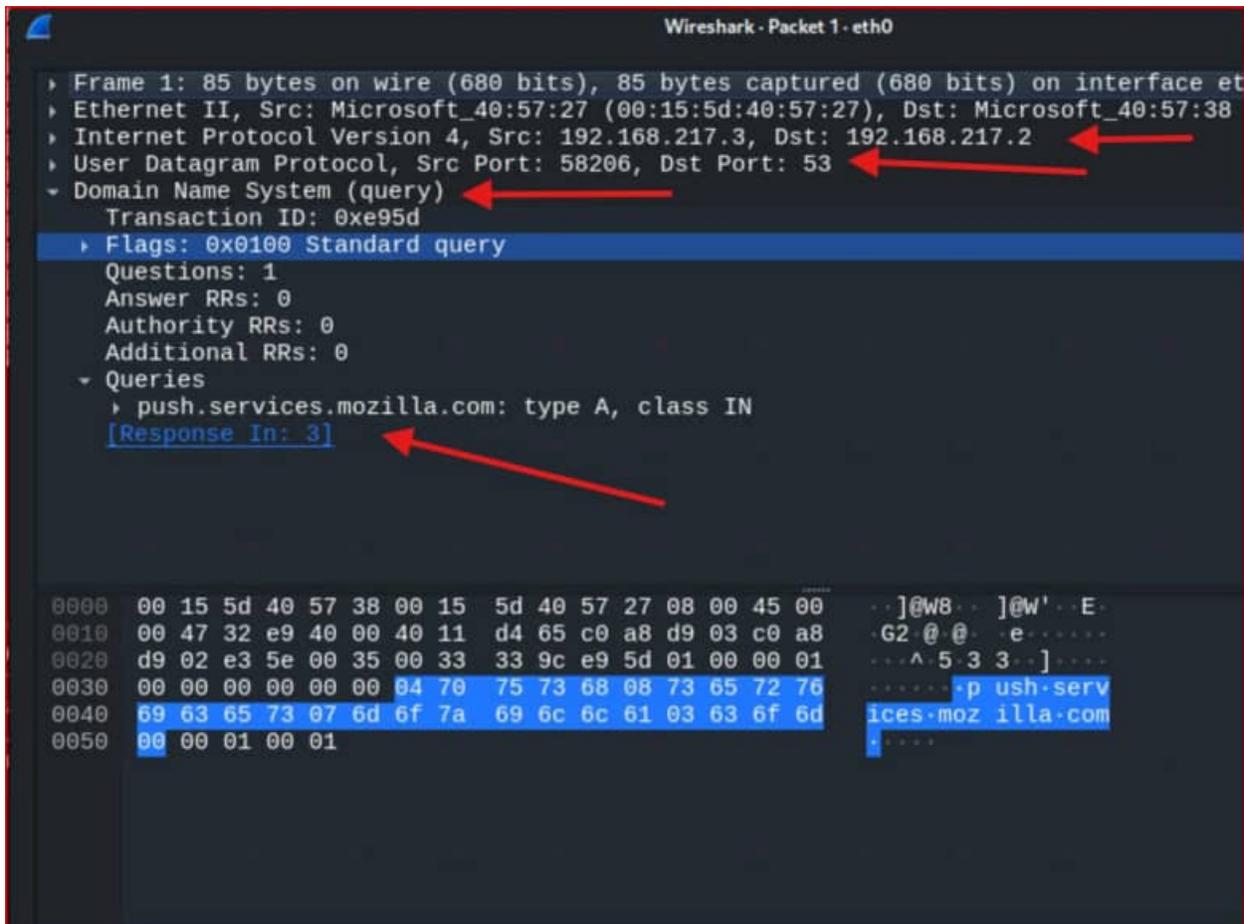
Screenprint:



Q5. (5 pts): Find a DNS query packet. What is the **domain name** this host is trying to resolve? What is the **source IP** and **port number**, **destination IP** and **port number**?

Answer: **Domain name** the host is trying to resolve is **push.services.mozilla.com**. The **source IP** is **192.168.217.3** and the **source port number** is **58206**. The **destination IP** is **192.168.217.2** and the **destination port number** is **53**.

Screenprint:



The screenshot shows a Wireshark packet capture of a DNS query. The packet list pane shows the following details:

- Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface eth0
- Ethernet II, Src: Microsoft_40:57:27 (00:15:5d:40:57:27), Dst: Microsoft_40:57:38
- Internet Protocol Version 4, Src: 192.168.217.3, Dst: 192.168.217.2
- User Datagram Protocol, Src Port: 58206, Dst Port: 53
- Domain Name System (query) Transaction ID: 0xe95d
- Flags: 0x0100 Standard query
- Questions: 1
- Answer RRs: 0
- Authority RRs: 0
- Additional RRs: 0
- Queries
 - push.services.mozilla.com: type A, class IN

The packet bytes pane shows the raw data of the packet, with the domain name 'push.services.mozilla.com' highlighted in blue. The hex and ASCII representations are as follows:

```
0000  00 15 5d 40 57 38 00 15 5d 40 57 27 08 00 45 00  ..]@w8.. ]@w'..E.
0010  00 47 32 e9 40 00 40 11 d4 65 c0 a8 d9 03 c0 a8  .G2.@.@.e.....
0020  d9 02 e3 5e 00 35 00 33 33 9c e9 5d 01 00 00 01  ...^5.33...
0030  00 00 00 00 00 00 04 70 75 73 68 08 73 65 72 76  ....p ush.serv
0040  69 63 65 73 07 6d 6f 7a 69 6c 6c 61 03 63 6f 6d  ices.moz illa.com
0050  00 00 01 00 01  .....
```

Q6. (5 pts): Find the **corresponding** (i.e., **click Response In: #**) DNS response to the query you selected at the previous step, and what is the source IP and port number, destination IP and port number? What is the message replied from the DNS server?

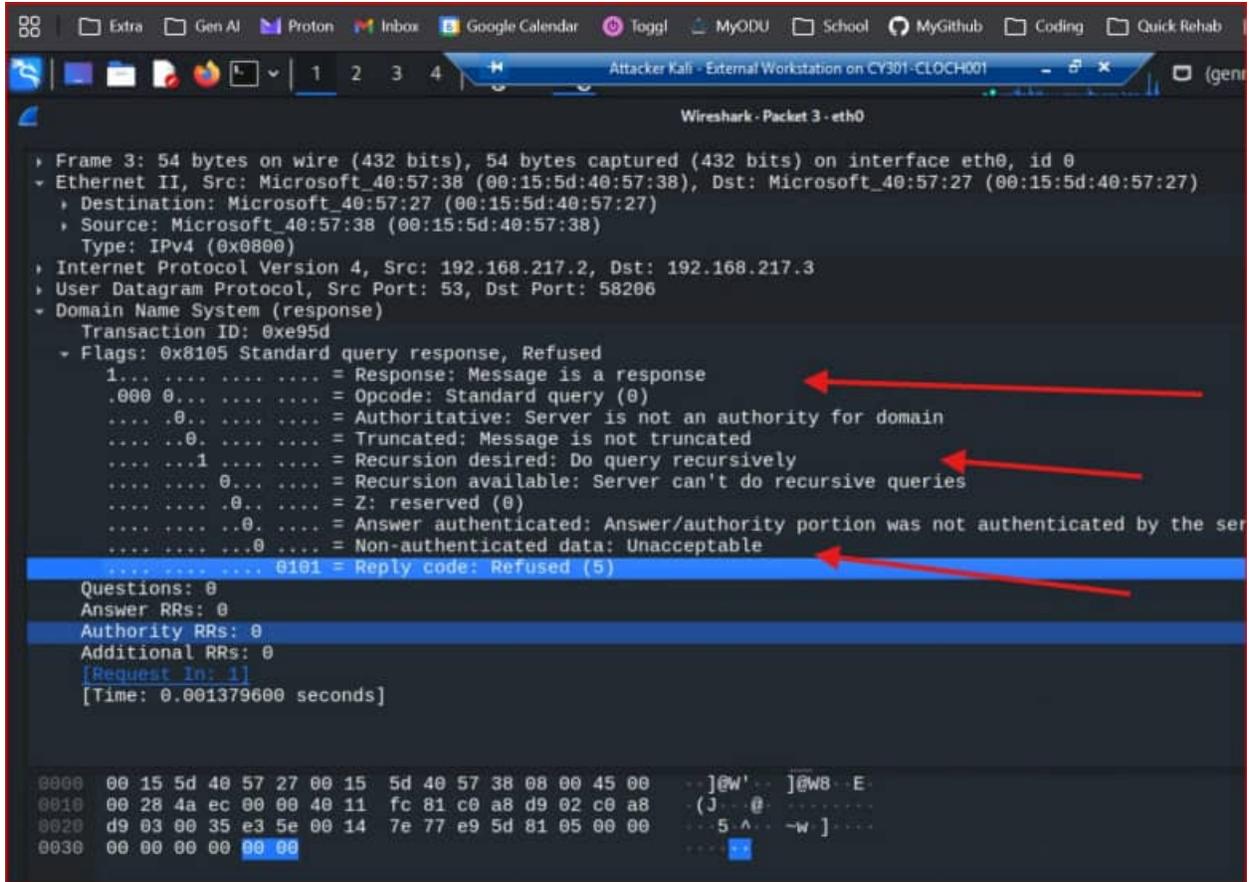
Answer: The **source IP is 192.168.217.2** and the **source port number is 53**. The **destination IP is 192.168.217.3** and the **destination port number is 58206**. Lastly, the DNS server replied with "**Reply code: Refused (5)**". The server refused to process the DNS query for `push.services.mozilla.com`. The response indicates that while the server received the query, it declined to provide an answer. There are **no answer records (Answer RRs: 0)** in the response because the **query was refused rather than resolved**.

Screenprint:

The screenshot shows a Wireshark capture of network traffic on the 'dns' filter. The packet list pane shows several DNS packets. Packet 3 is selected, which is a 'Standard query response' from 192.168.217.2 to 192.168.217.3. The packet details pane for this packet shows the following structure:

- Internet Protocol Version 4, Src: 192.168.217.2, Dst: 192.168.217.3
- User Datagram Protocol, Src Port: 53, Dst Port: 58206
- Domain Name System (response)
 - Transaction ID: 0xe95d
 - Flags: 0xB105 Standard query response, Refused
 - Questions: 0
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 0

Red arrows in the image point to the source IP (192.168.217.2) and source port (53) in the IP and UDP sections, and the destination IP (192.168.217.3) and destination port (58206) in the UDP section.



Task B: Sniff LAN traffic (70 Points)

In this task, you will be acting as an **ATTACKER** who sniffs the regular communications between peers (External Attacker Kali and Ubuntu) by using Wireshark on **Internal Attacker Kali VM**.

I would recommend you keep the Wireshark running on Internal Kali all the time.

IMPORTANT NOTES!

* Because the current Hyper-V setting in CCIA does not “broadcast” the communication between hosts in the same network, we need to enable port mirroring to allow Internal Kali to “see” communications between External Kali and Ubuntu.

To be specific, you need to put the sniffer (Internal Kali) as the **mirroring Destination**, and the target VMs are **mirroring Source** (See the above Figure 1 under Task-B).

Each VM has two network adapters, one for regular connection and the other for sharing with the CCIA server. We need to configure port mirroring on the first adapter. To be specific,

- Internal Kali: Set Mirroring mode to “**Destination**” in the “**Port Mirroring**”
- Ubuntu Kali: Set Mirroring mode to “**Source**” in the “**Port Mirroring**”
- External Kali: Set Mirroring mode to “**Source**” in the “**Port Mirroring**”

Here is the screenshot showing how to enable Port Mirroring to “source” in External Kali, for your reference. You need to set for Ubuntu and Internal Kali as asked in the instructions above and complete the steps.

1. Sniff ICMP traffic (5 + 5 + 5 + 15 = 30 points)

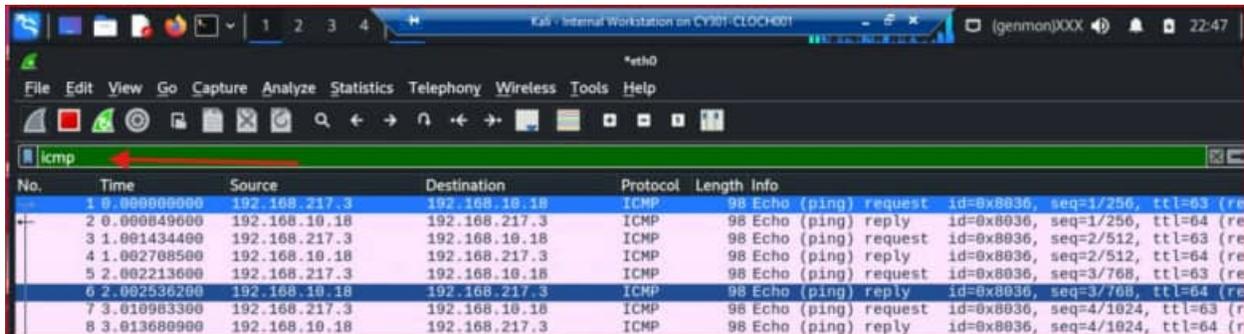
- Please turn on Attacker/External Kali, internal kali, pfsense, and Ubuntu.

Answer: All four necessary VMs are running.

| Name | State | CPU Usage | Assigned Memory | Uptime | Status | Configurati... |
|--------------------------------|---------|-----------|-----------------|----------|--------|----------------|
| Attacker Kali - External Wo... | Running | 7% | 3072 MB | 00:06:40 | | 9.0 |
| Kali - Internal Workstation | Running | 33% | 3072 MB | 00:06:31 | | 9.0 |
| pFsense - Firewall 64 2.7.2 | Running | 0% | 1024 MB | 02:46:53 | | 9.0 |
| Ubuntu 2204-64-bit | Running | 2% | 2048 MB | 00:06:22 | | 9.0 |
| Windows 10 | Off | | | | | 9.0 |
| Windows 7 | Off | | | | | 9.0 |
| Windows Server 2022 | Off | | | | | 9.0 |
| Windows XP Professional | Off | | | | | 9.0 |

- d. Apply proper display or capture filter in Wireshark on **Internal Kali VM** to show active ICMP traffic.

Answer: Filter: icmp

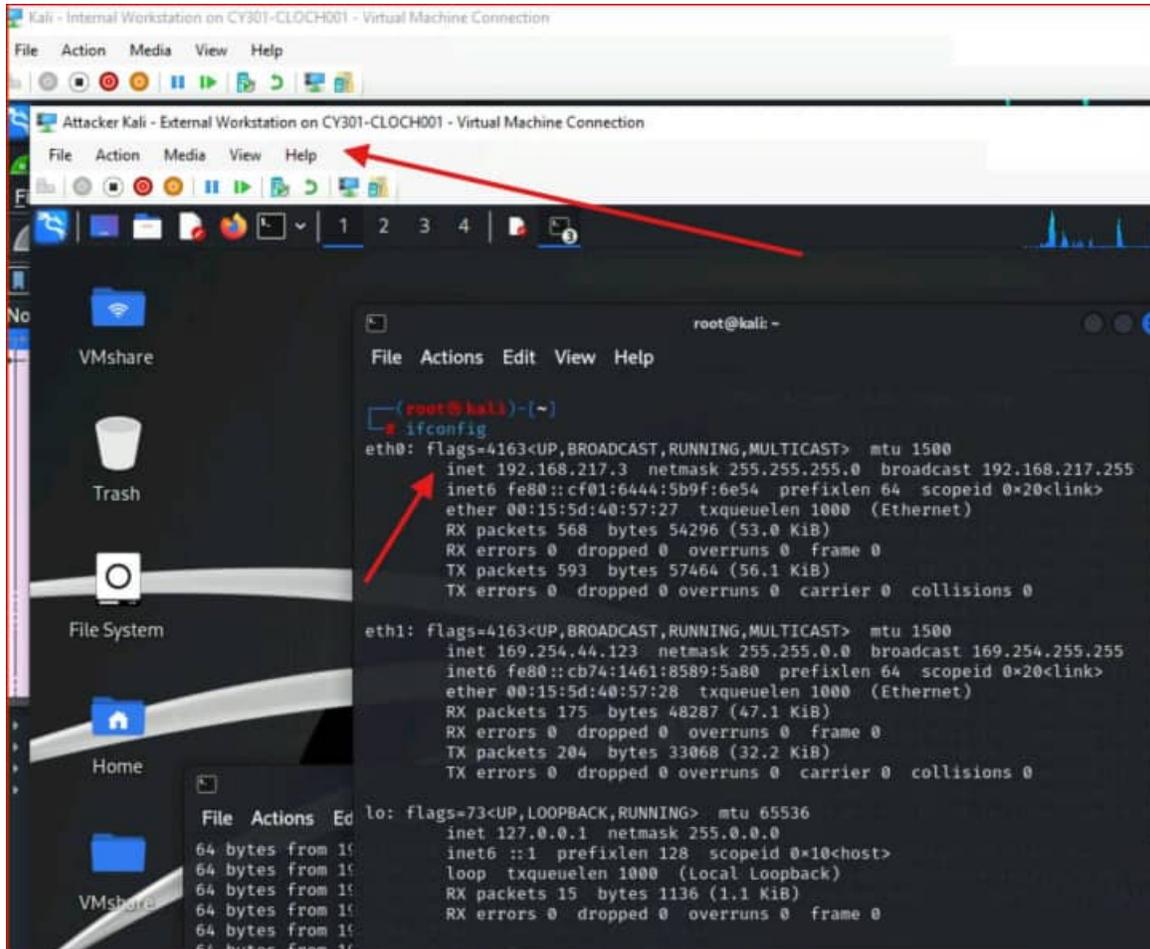


The screenshot shows the Wireshark interface with the display filter set to 'icmp'. The packet list pane shows the following data:

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|---------------|---------------|----------|--------|--|
| 1 | 0.000000000 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=1/256, ttl=63 (re |
| 2 | 0.000049600 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8036, seq=1/256, ttl=64 (re |
| 3 | 1.001434400 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=2/512, ttl=63 (re |
| 4 | 1.002708500 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8036, seq=2/512, ttl=64 (re |
| 5 | 2.002213600 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=3/768, ttl=63 (re |
| 6 | 2.002536200 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8036, seq=3/768, ttl=64 (re |
| 7 | 3.010903300 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=4/1024, ttl=63 (r |
| 8 | 3.013680900 | 192.168.10.18 | 192.168.217.3 | ICMP | 98 | Echo (ping) reply id=0x8036, seq=4/1024, ttl=64 (r |

- e. Apply a proper display or capture filter on the internal Kali VM that **ONLY** displays the ICMP request that originated from the external Kali VM and goes to the Ubuntu 64-bit VM.

Answer: *Filter:* `icmp.type == 8 && ip.src == 192.168.217.3 && ip.dst == 192.168.10.18`

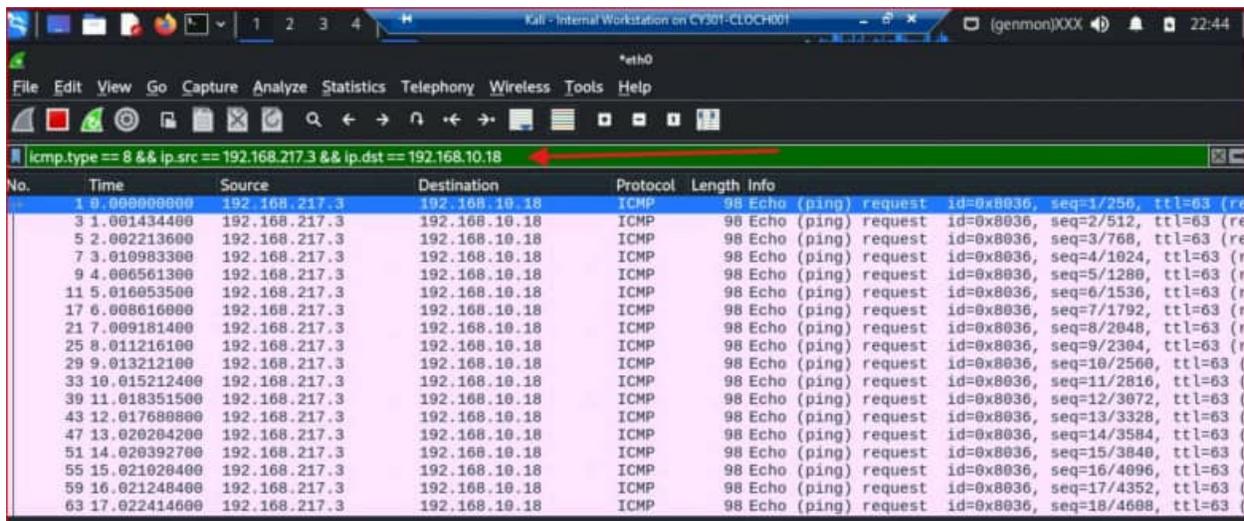


```

root@kali: ~
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.217.3 netmask 255.255.255.0 broadcast 192.168.217.255
    inet6 fe80::cf01:6444:5b9f:6e54 prefixlen 64 scopeid 0<20<link>
    ether 00:15:5d:40:57:27 txqueuelen 1000 (Ethernet)
    RX packets 568 bytes 54296 (53.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 593 bytes 57464 (56.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 169.254.44.123 netmask 255.255.0.0 broadcast 169.254.255.255
    inet6 fe80::cb74:1461:8589:5a80 prefixlen 64 scopeid 0<20<link>
    ether 00:15:5d:40:57:28 txqueuelen 1000 (Ethernet)
    RX packets 175 bytes 48287 (47.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 204 bytes 33068 (32.2 KiB)
    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 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 15 bytes 1136 (1.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
  
```

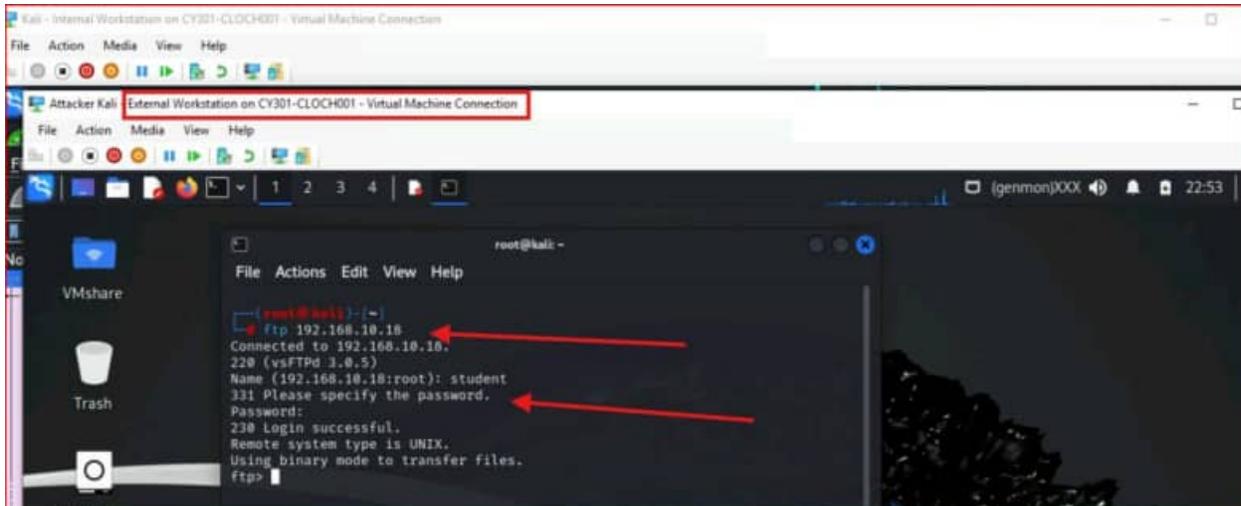


| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|---------------|---------------|----------|--------|--|
| 1 | 0.000000000 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=1/250, ttl=63 (re |
| 3 | 1.001434400 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=2/512, ttl=63 (re |
| 5 | 2.002213600 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=3/768, ttl=63 (re |
| 7 | 3.010983300 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=4/1024, ttl=63 (r |
| 9 | 4.006561300 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=5/1280, ttl=63 (r |
| 11 | 5.016053500 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=6/1536, ttl=63 (r |
| 17 | 6.008616000 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=7/1792, ttl=63 (r |
| 21 | 7.009181400 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=8/2048, ttl=63 (r |
| 25 | 8.011216100 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=9/2304, ttl=63 (r |
| 29 | 9.013212100 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=10/2560, ttl=63 (|
| 33 | 10.015212400 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=11/2816, ttl=63 (|
| 39 | 11.018351500 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=12/3072, ttl=63 (|
| 43 | 12.017680800 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=13/3328, ttl=63 (|
| 47 | 13.020204200 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=14/3584, ttl=63 (|
| 51 | 14.020392700 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=15/3840, ttl=63 (|
| 55 | 15.021020400 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=16/4096, ttl=63 (|
| 59 | 16.021248400 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=17/4352, ttl=63 (|
| 63 | 17.022414600 | 192.168.217.3 | 192.168.10.18 | ICMP | 98 | Echo (ping) request id=0x8036, seq=18/4608, ttl=63 (|

2. Sniff FTP traffic (10 + 15 + 15 = 40 pts points)

- a. **Ubuntu VM** is also serving as an FTP server inside the LAN network. Now, you need to use External Kali to access this FTP server by using the command: **ftp [ip_addr of ubuntu VM]**. The username for the FTP server is **student**, and the password is **password**.

Answer: **ftp 192.168.10.18**. I connected to the FTP server running on Ubuntu VM using the command '**ftp 192.168.10.18**' from External Kali. I logged in with **username 'student'** and **password 'password'**, which returned a '230 Login successful' message.



- b. **Unfortunately**, Internal Kali, the attacker, is also sniffing into the communication. Therefore, all of your communication is exposed to the attacker. Now, you need to find out the **password** used by External Kali to access the FTP server from the intercepted traffic on Internal Kali. You need to take a screenshot and explain how you found the password.

Answer: As an attacker on Internal Kali VM, I intercepted the FTP traffic using Wireshark with the '**ftp**' display filter. I found the **username** in a packet containing '**USER student**' and the **password** in a packet containing '**PASS password**'. This demonstrates that FTP transmits credentials in **plaintext**, making them easily accessible to anyone monitoring network traffic. Again, the username is 'student' and the password is 'password;' the login was successful.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|---------------|---------------|---------------|----------|--------|--|
| 134 | 700.488656000 | 192.168.10.18 | 192.168.217.3 | FTP | 86 | Response: 220 (vsFTPd 3.0.5) |
| 138 | 718.177215100 | 192.168.217.3 | 192.168.10.18 | FTP | 86 | Request: USER student |
| 140 | 718.178177800 | 192.168.10.18 | 192.168.217.3 | FTP | 100 | Response: 331 Please specify the password. |
| 142 | 720.832508500 | 192.168.217.3 | 192.168.10.18 | FTP | 81 | Request: PASS password |
| 143 | 720.864977700 | 192.168.10.18 | 192.168.217.3 | FTP | 89 | Response: 230 Login successful. |
| 145 | 720.868908400 | 192.168.217.3 | 192.168.10.18 | FTP | 72 | Request: SYST |
| 146 | 720.870532600 | 192.168.10.18 | 192.168.217.3 | FTP | 85 | Response: 215 UNIX Type: LB |
| 147 | 720.879912500 | 192.168.217.3 | 192.168.10.18 | FTP | 72 | Request: FEAT |
| 148 | 720.879926400 | 192.168.10.18 | 192.168.217.3 | FTP | 81 | Response: 211-Features: |
| 149 | 720.879927800 | 192.168.10.18 | 192.168.217.3 | FTP | 87 | Response: EPRT |
| 150 | 720.879935400 | 192.168.10.18 | 192.168.217.3 | FTP | 110 | Response: PASV |

- c. After you successfully find the username & password from the FTP traffic, repeat the previous step (2.a), and use your **MIDAS ID** as the username and **UIN** as the password to re-access the FTP server from External Kali. Although External Kali may not access the FTP server, you need to intercept the packets containing these “secrets” from the attacker VM, which is **Internal Kali**.

Answer: The username is 'cloch001', the UIN is 01290201. Login unsuccessful.

