Live Packet Sniffing Exercise

Premise

- An unknown party is sending malicious packets to the users network, the user needs to figure out the payload of these packets
  - NIST: K0062, K0301

Student Steps

- Log into Windows machine
  - Enter **CMD** and type *ipconifg*
  - Look for **IPv4 Address** (10.16.1.XYZ)
- Log into Linux Machine
  - Open a **terminal** on the Desktop and run "./PacketSend WINDOWS_IPv4_Address"
- Log into Windows Machine
  - Complete exercise
- Return to Linux machine and close the terminal

Recommend Tools

- Wireshark

Questions

- Which *protocol* are the malicious packets using
- What *port* are the packets targeting
- What is text/ASCII *payload* of the malicious packet (data segment)

FILE

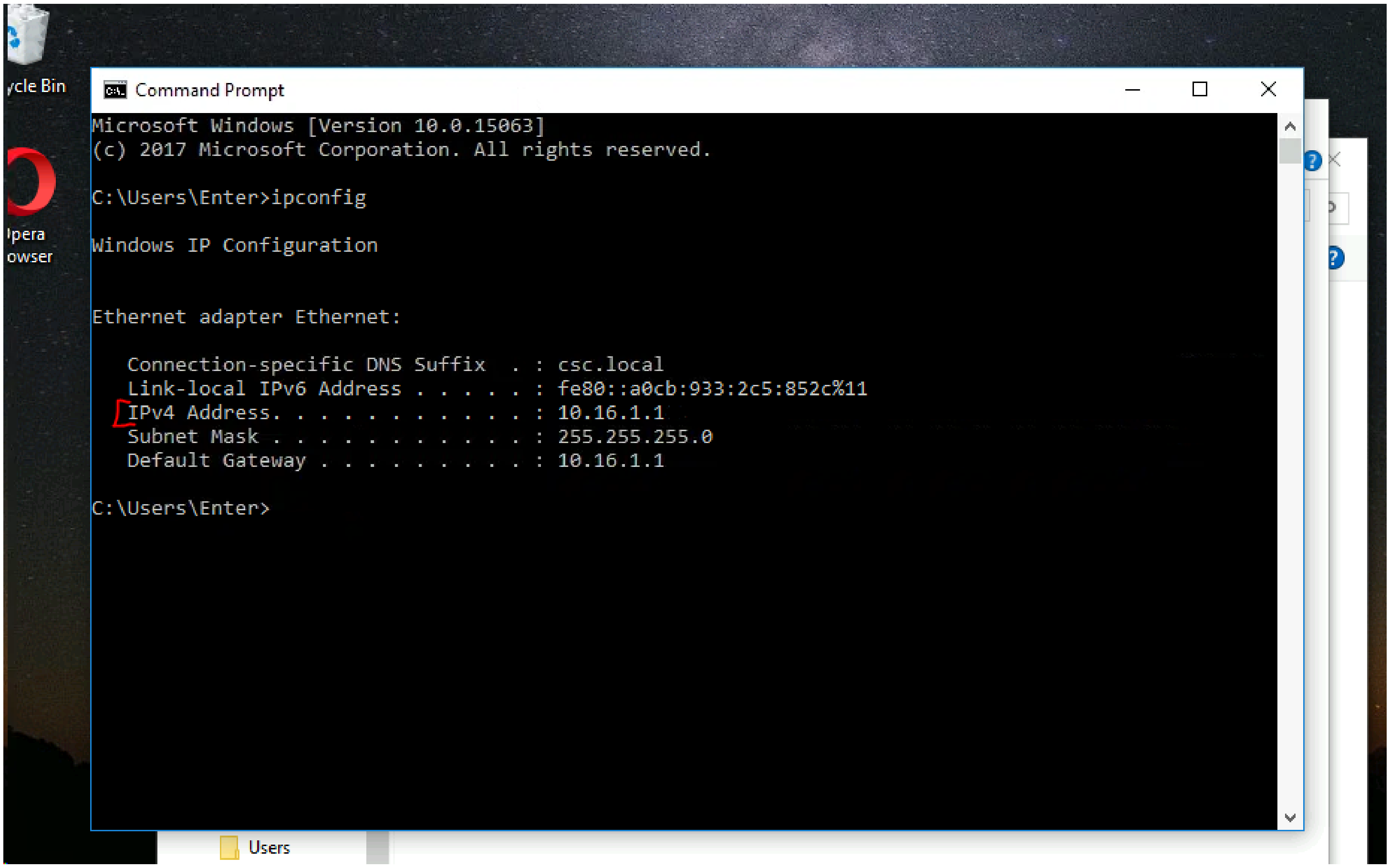
- PacketSend

Answers

- Which *protocol* are the malicious packets using
    - UDP
- What *port* are the packets targeting
    - 2025
- What is text/ASCII *payload* of the malicious packet (data segment)
    - "0x7E9 Was Here..."

Walkthrough

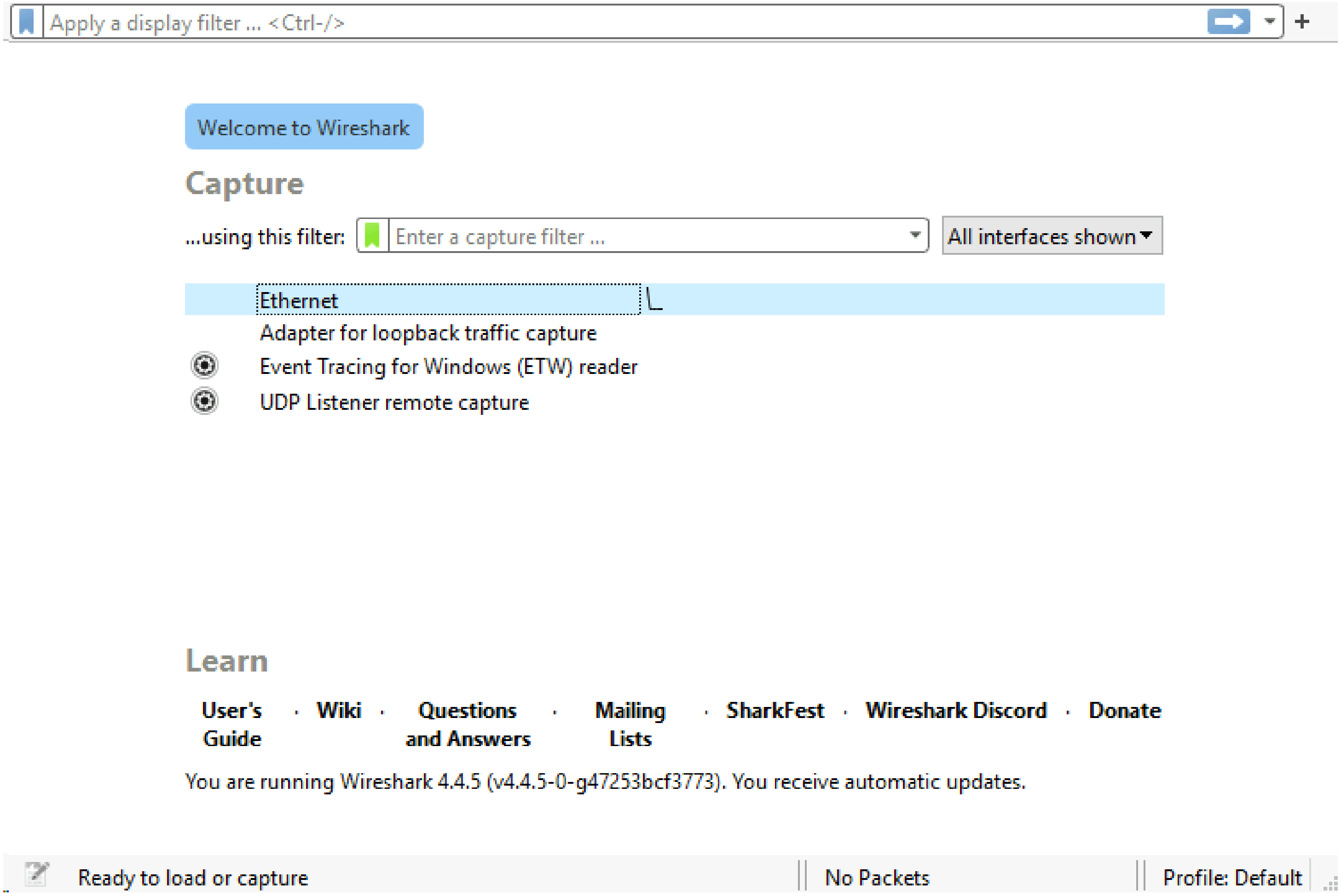
1. Log into the Winodws VDI machine
2. Open a command prompt window and enter *ipconfig*



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Enter>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : csc.local
   Link-local IPv6 Address . . . . . : fe80::a0cb:933:2c5:852c%11
   IPv4 Address. . . . . . . . . . . : 10.16.1.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.16.1.1

C:\Users\Enter>
```

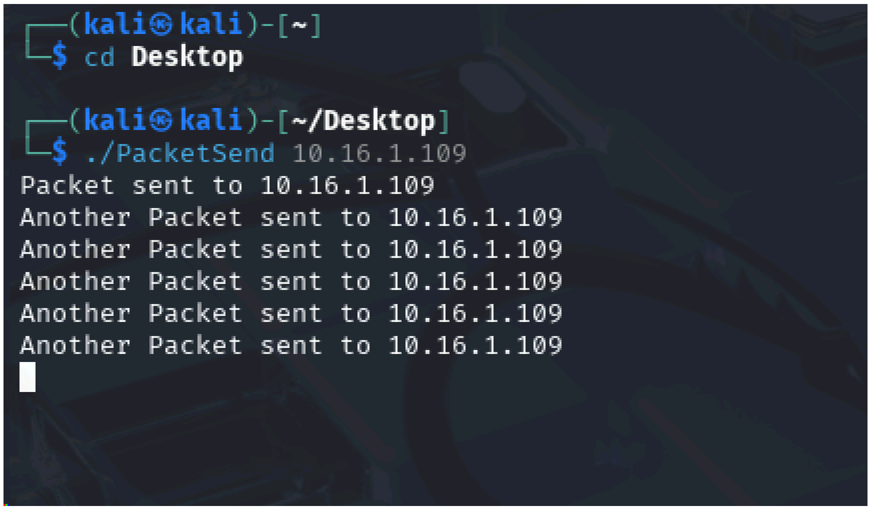
    1. Note down the Ipv4 address that was provided

3. Open Wireshark on Windows, Double click on *Ethernet* to view all packets



You are running Wireshark 4.4.5 (v4.4.5-0-g47253bcf3773). You receive automatic updates.

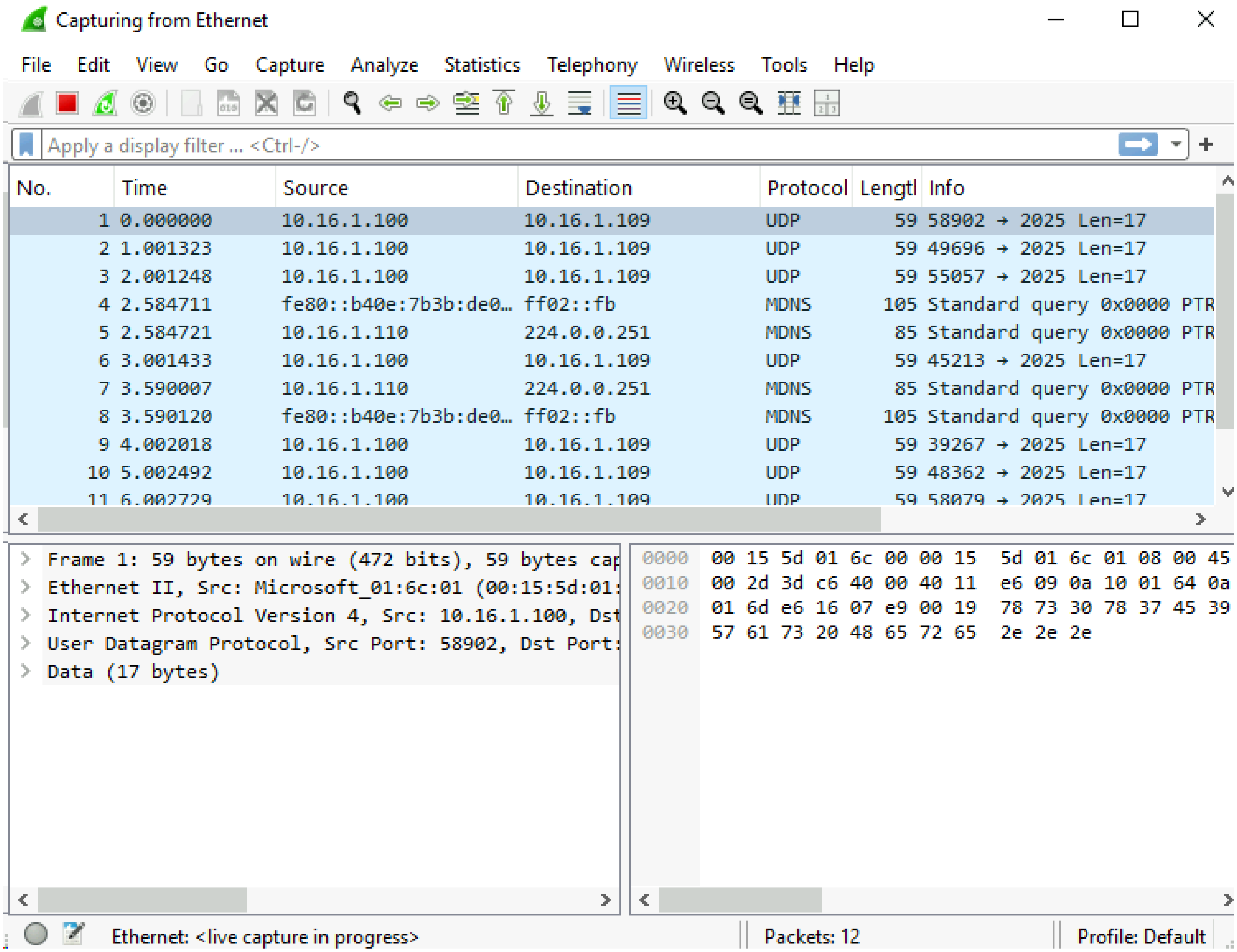
4. Log into the linux machine and run the *PacketSend* file with the windows machine IP address



```
┌──(kali㉿kali)-[~]
└─$ cd Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ ./PacketSend 10.16.1.109
Packet sent to 10.16.1.109
Another Packet sent to 10.16.1.109
Another Packet sent to 10.16.1.109
Another Packet sent to 10.16.1.109
Another Packet sent to 10.16.1.109
Another Packet sent to 10.16.1.109
```

5. Return to the Windows machine to start the packet analysis

6. Packets should now be coming in from the Attacker machine



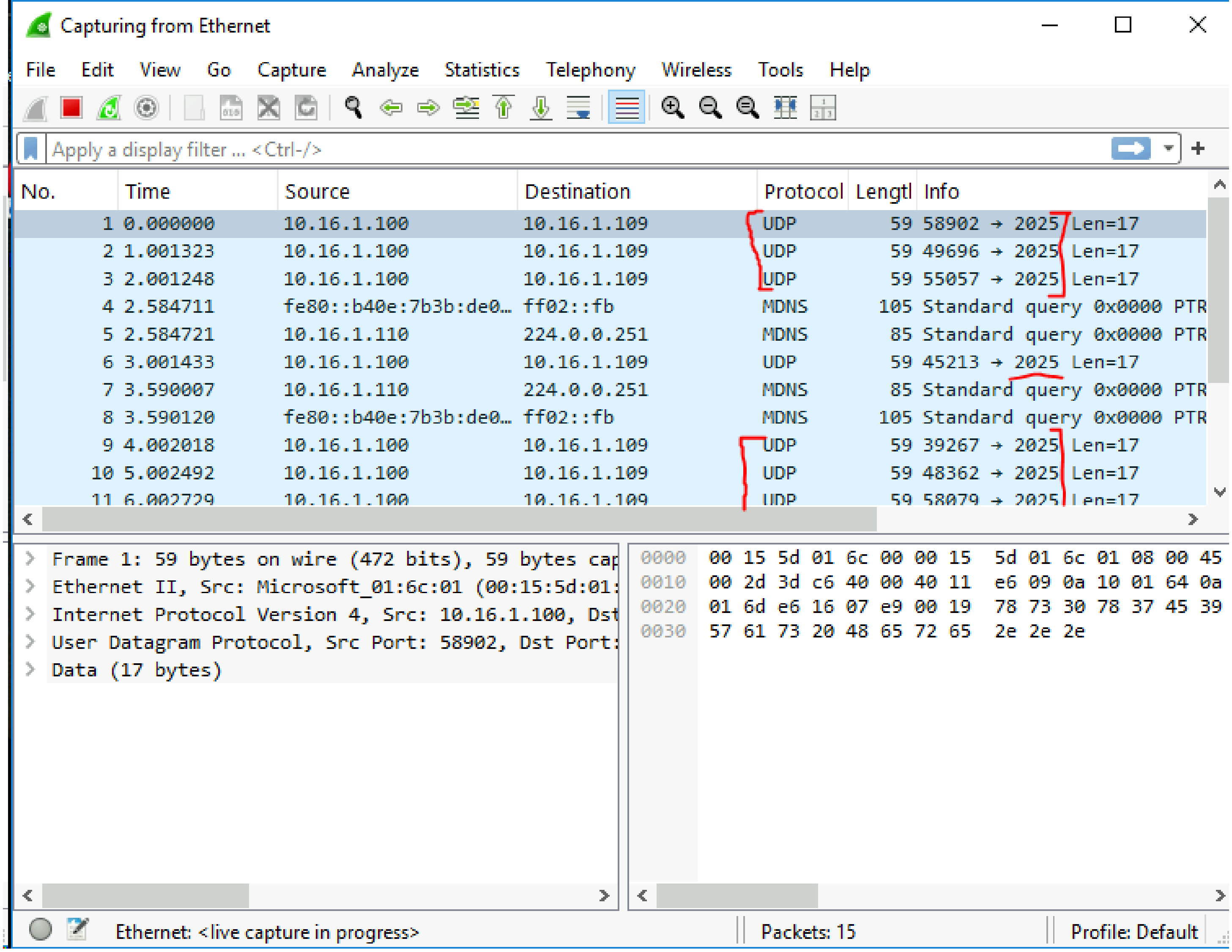
```
No.   Time       Source                 Destination   Protocol  Lengtl Info
    1 0.000000   10.16.1.100            10.16.1.109   UDP          59 58902 → 2025 Len=17
    2 1.001323   10.16.1.100            10.16.1.109   UDP          59 49696 → 2025 Len=17
    3 2.001248   10.16.1.100            10.16.1.109   UDP          59 55057 → 2025 Len=17
    4 2.584711   fe80::b40e:7b3b:de0…   ff02::fb      MDNS        105 Standard query 0x0000 PTR
    5 2.584721   10.16.1.110            224.0.0.251   MDNS         85 Standard query 0x0000 PTR
    6 3.001433   10.16.1.100            10.16.1.109   UDP          59 45213 → 2025 Len=17
    7 3.590007   10.16.1.110            224.0.0.251   MDNS         85 Standard query 0x0000 PTR
    8 3.590120   fe80::b40e:7b3b:de0…   ff02::fb      MDNS        105 Standard query 0x0000 PTR
    9 4.002018   10.16.1.100            10.16.1.109   UDP          59 39267 → 2025 Len=17
   10 5.002492   10.16.1.100            10.16.1.109   UDP          59 48362 → 2025 Len=17
   11 6.002729   10.16.1.100            10.16.1.109   UDP          59 58079 → 2025 Len=17
```

```
> Frame 1: 59 bytes on wire (472 bits), 59 bytes cap
> Ethernet II, Src: Microsoft_01:6c:01 (00:15:5d:01:
> Internet Protocol Version 4, Src: 10.16.1.100, Dst
> User Datagram Protocol, Src Port: 58902, Dst Port:
> Data (17 bytes)
```

```
0000  00 15 5d 01 6c 00 00 15  5d 01 6c 01 08 00 45
0010  00 2d 3d c6 40 00 40 11  e6 09 0a 10 01 64 0a
0020  01 6d e6 16 07 e9 00 19  78 73 30 78 37 45 39
0030  57 61 73 20 48 65 72 65  2e 2e 2e
```

Ethernet: <live capture in progress> | Packets: 12 | Profile: Default

7. Notice the Protocol is **UDP** under the protocol column, and that the target port is **2025** in the info column



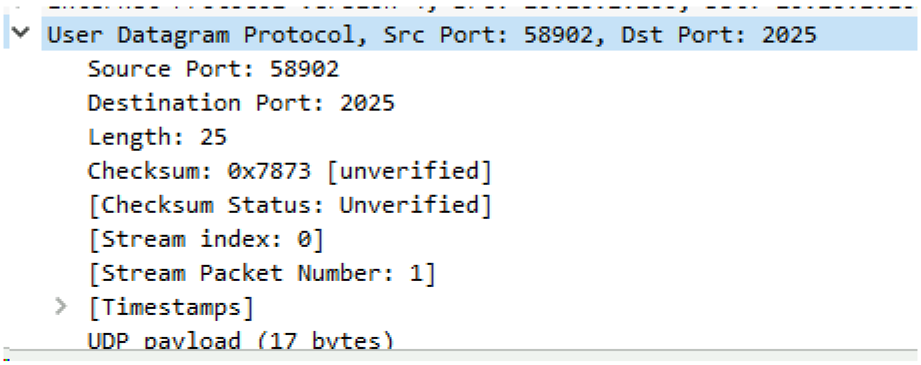
```
No.   Time       Source                  Destination   Protocol  Lengtl Info
    1 0.000000   10.16.1.100             10.16.1.109   UDP           59 58902 → 2025 Len=17
    2 1.001323   10.16.1.100             10.16.1.109   UDP           59 49696 → 2025 Len=17
    3 2.001248   10.16.1.100             10.16.1.109   UDP           59 55057 → 2025 Len=17
    4 2.584711   fe80::b40e:7b3b:de0…    ff02::fb      MDNS         105 Standard query 0x0000 PTR
    5 2.584721   10.16.1.110             224.0.0.251   MDNS          85 Standard query 0x0000 PTR
    6 3.001433   10.16.1.100             10.16.1.109   UDP           59 45213 → 2025 Len=17
    7 3.590007   10.16.1.110             224.0.0.251   MDNS          85 Standard query 0x0000 PTR
    8 3.590120   fe80::b40e:7b3b:de0…    ff02::fb      MDNS         105 Standard query 0x0000 PTR
    9 4.002018   10.16.1.100             10.16.1.109   UDP           59 39267 → 2025 Len=17
   10 5.002492   10.16.1.100             10.16.1.109   UDP           59 48362 → 2025 Len=17
   11 6.002729   10.16.1.100             10.16.1.109   UDP           59 58079 → 2025 Len=17
```

```
> Frame 1: 59 bytes on wire (472 bits), 59 bytes cap
> Ethernet II, Src: Microsoft_01:6c:01 (00:15:5d:01:
> Internet Protocol Version 4, Src: 10.16.1.100, Dst
> User Datagram Protocol, Src Port: 58902, Dst Port:
> Data (17 bytes)
```

```
0000  00 15 5d 01 6c 00 00 15  5d 01 6c 01 08 00 45
0010  00 2d 3d c6 40 00 40 11  e6 09 0a 10 01 64 0a
0020  01 6d e6 16 07 e9 00 19  78 73 30 78 37 45 39
0030  57 61 73 20 48 65 72 65  2e 2e 2e
```

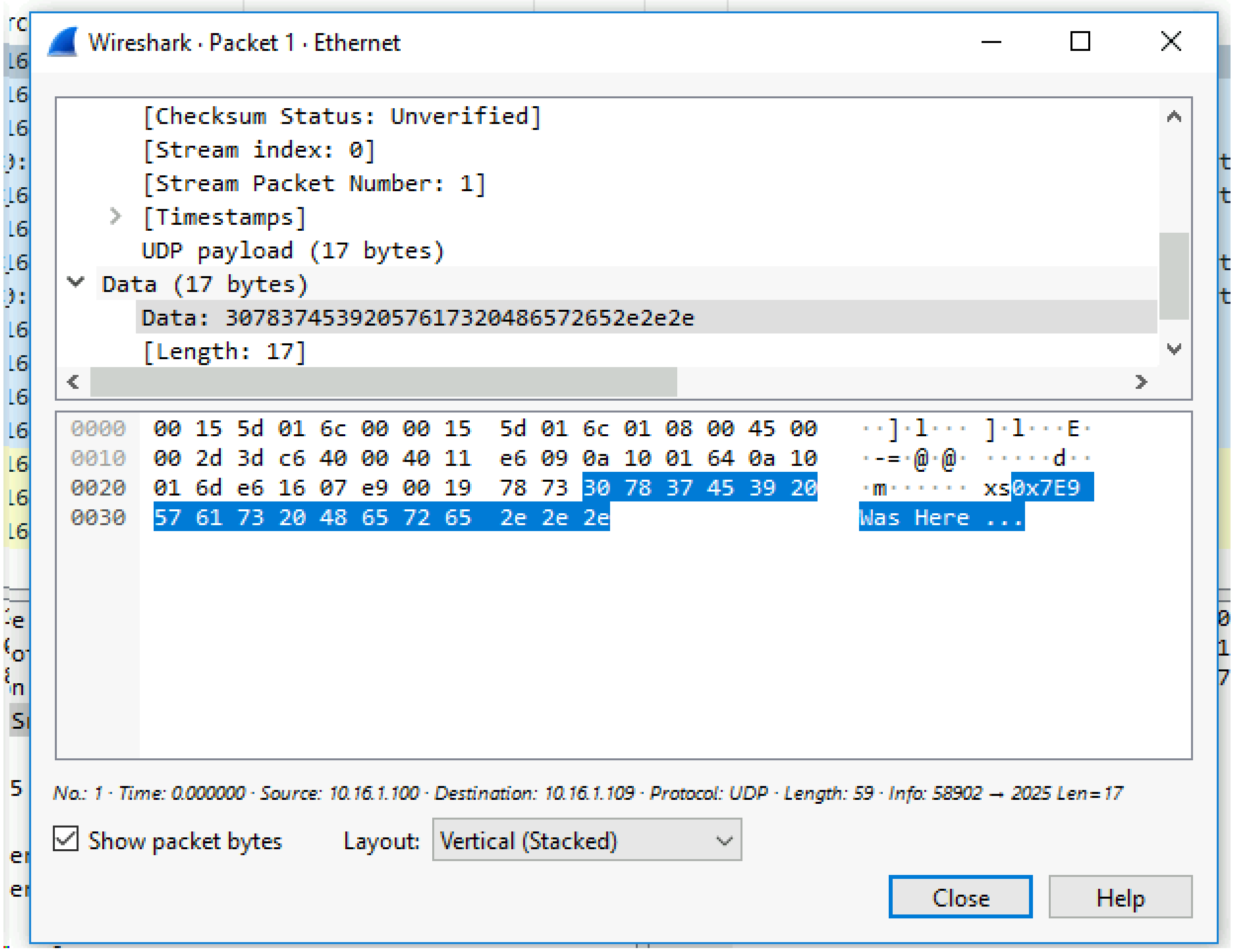
Ethernet: <live capture in progress>
Packets: 15
Profile: Default

1. A more comprehensive breakdown is found below



```
v User Datagram Protocol, Src Port: 58902, Dst Port: 2025
      Source Port: 58902
      Destination Port: 2025
      Length: 25
      Checksum: 0x7873 [unverified]
      [Checksum Status: Unverified]
      [Stream index: 0]
      [Stream Packet Number: 1]
    > [Timestamps]
      UDP payload (17 bytes)
```

8. To view the ASCII contents **Either** double click on any UDP packet from the attacking machine



Wireshark · Packet 1 · Ethernet

```
        [Checksum Status: Unverified]
        [Stream index: 0]
        [Stream Packet Number: 1]
      > [Timestamps]
        UDP payload (17 bytes)
  v  Data (17 bytes)
        Data: 3078374539205761732048657265 2e2e2e
        [Length: 17]
```

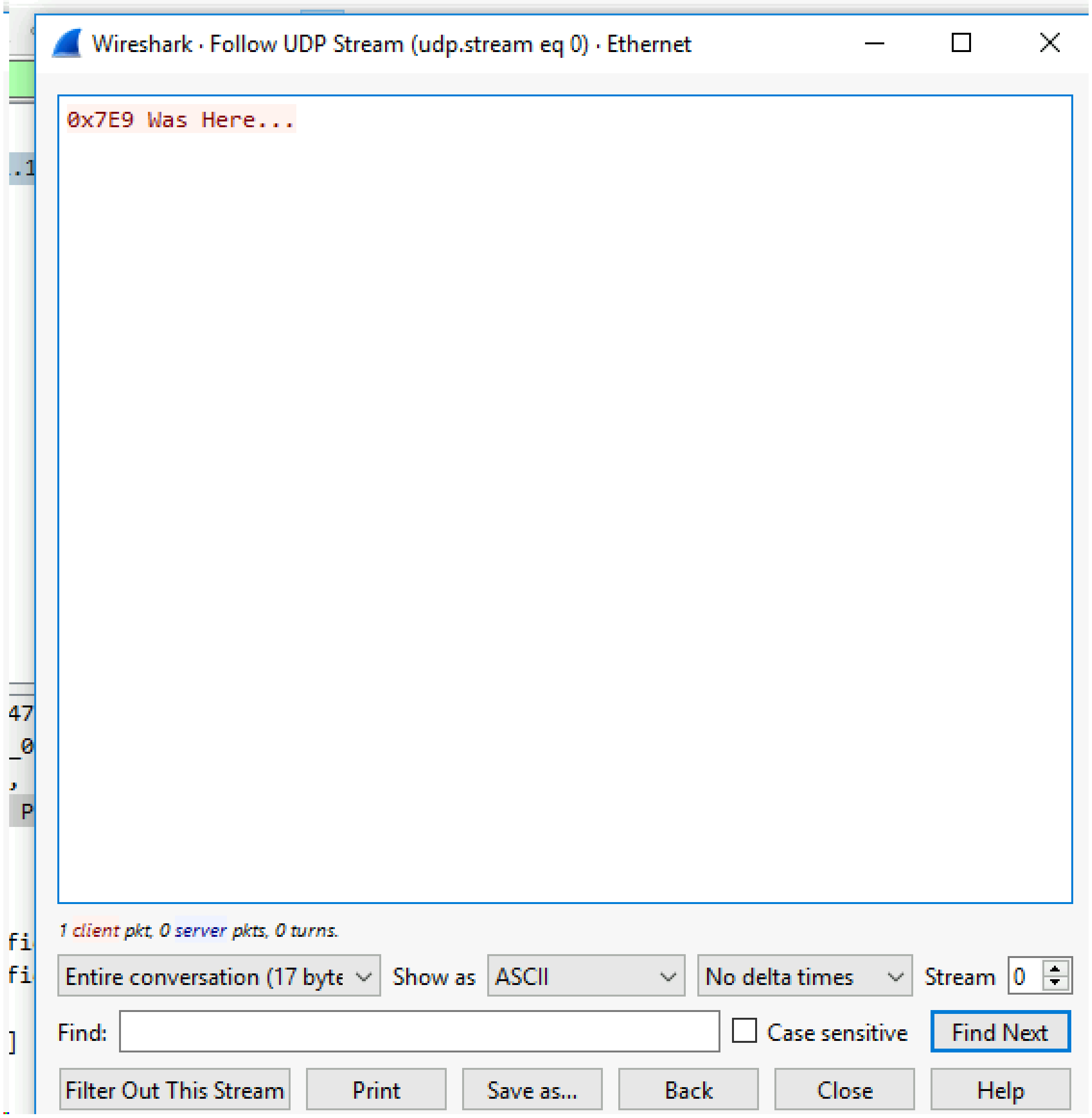
```
0000  00 15 5d 01 6c 00 00 15  5d 01 6c 01 08 00 45 00   ··]·l··· ]·l···E·
0010  00 2d 3d c6 40 00 40 11  e6 09 0a 10 01 64 0a 10   ·-=·@·@· ·····d··
0020  01 6d e6 16 07 e9 00 19  78 73 30 78 37 45 39 20   ·m······ xs0x7E9 
0030  57 61 73 20 48 65 72 65  2e 2e 2e                  Was Here ...
```

No.: 1 · Time: 0.000000 · Source: 10.16.1.100 · Destination: 10.16.1.109 · Protocol: UDP · Length: 59 · Info: 58902 → 2025 Len=17

☑ Show packet bytes

Layout: Vertical (Stacked)

9. **OR** right click on any UDP packet from the attacker machine and "*Follow UDP Stream*"



```
0x7E9 Was Here...
```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (17 byte | Show as ASCII | No delta times | Stream 0

10. It is highly recommended to spend some time reverse engineering this program! As it uses C sockets which are really cool!