Packet Capture and Analysis with Wireshark and Tcpdump

Overview

As a cybersecurity analyst trainee, I carried out hands-on exercises using **Wireshark** and **tcpdump** to analyze and capture network traffic. These activities helped me build skills in packet inspection, protocol analysis, and filtering network traffic to identify relevant information.

Tools & Technologies

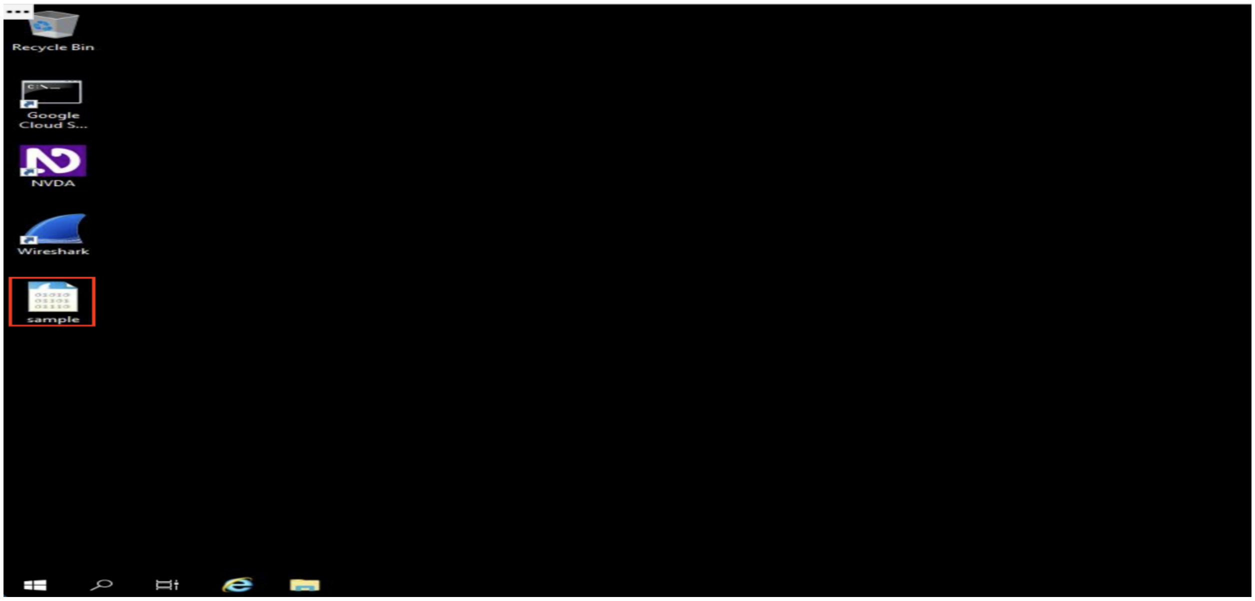
- **Wireshark** – GUI-based packet analyzer for detailed inspection of network traffic
- **tcpdump** – Command-line tool for live packet capture and filtering in Linux
- **PCAP files** – For storing and reviewing captured traffic

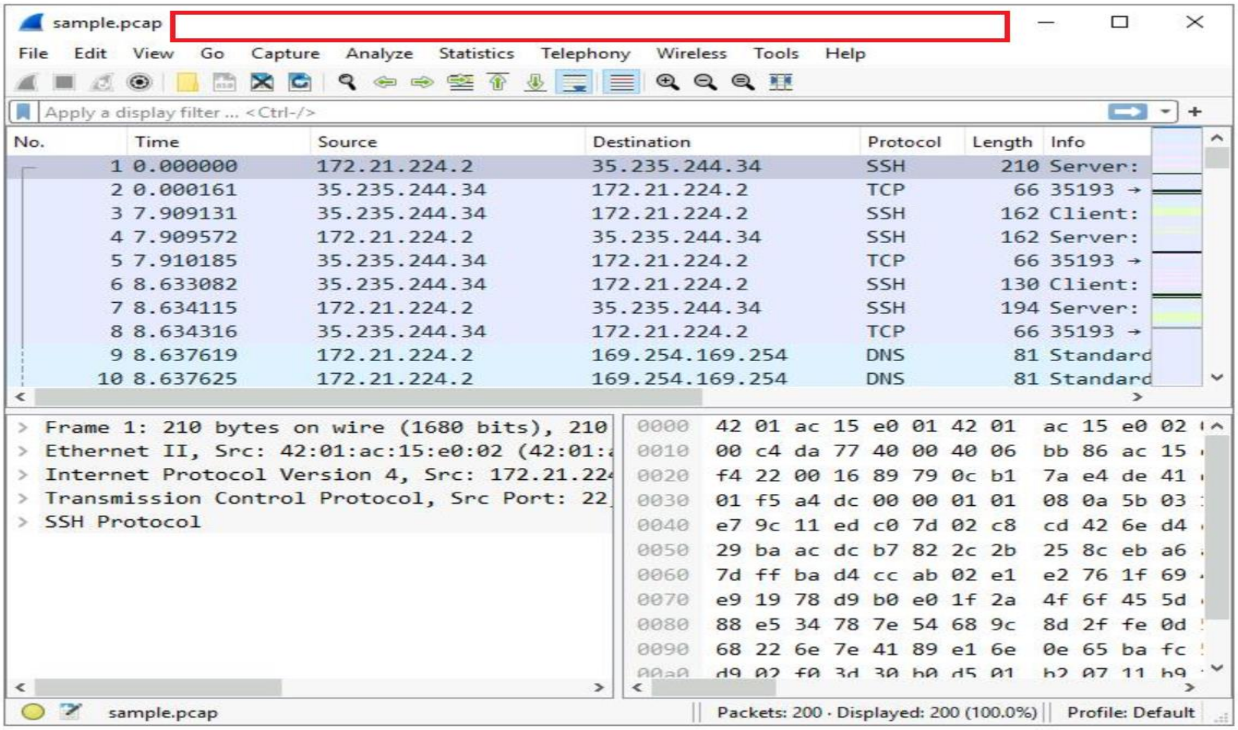
Key Activities & Screenshots

1. Analyzing Packets with Wireshark

- Opened and explored a `.pcap` file in Wireshark.
- Examined packet details including frame length, source and destination IP addresses, and protocols.
- Applied filters such as:
  - `ip.addr == <IP>` (filter by IP address)
  - `udp.port == 53` (filter DNS traffic)
  - `tcp.port == 80` (filter HTTP traffic)
  - `tcp contains "curl"` (search payload text).
- Identified communication protocols (e.g., ICMP, TCP, UDP, DNS).
- Verified DNS queries and responses (e.g., resolving `opensource.google.com` to IP `142.250.1.139`).
- Inspected TCP headers to analyze flags, sequence numbers, and ports.

Fig. 1. VM window




sample.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

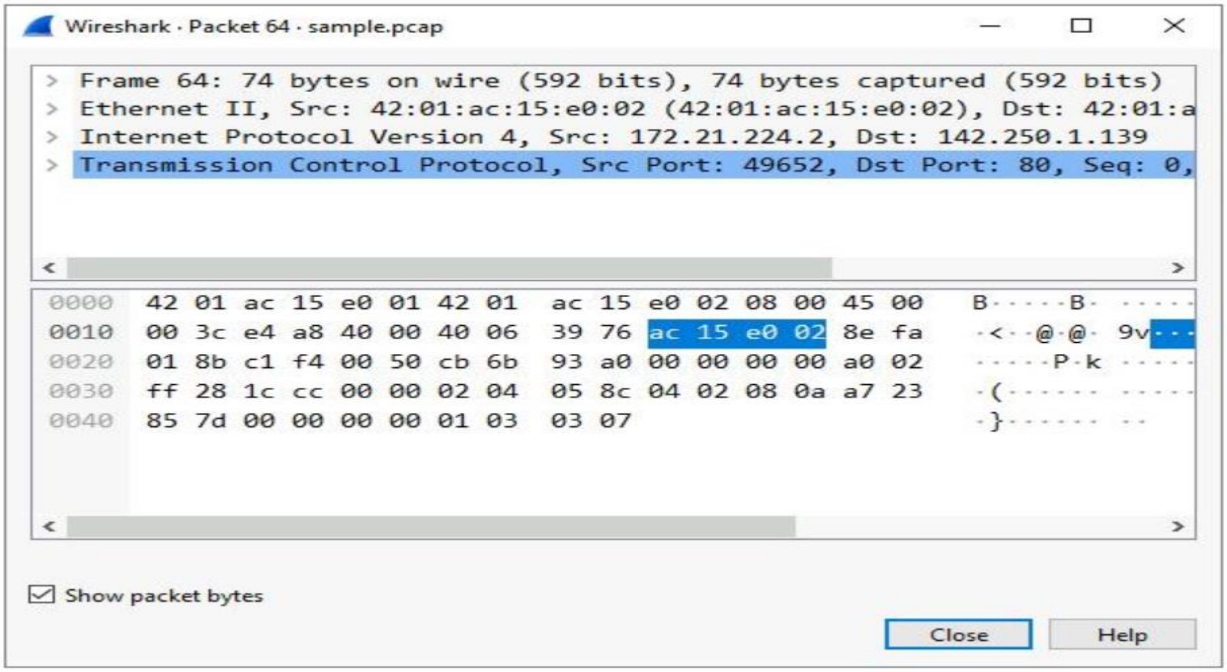
```
No.      Time        Source            Destination        Protocol  Length Info
      1 0.000000    172.21.224.2      35.235.244.34      SSH          210 Server:
      2 0.000161    35.235.244.34     172.21.224.2       TCP           66 35193 →
      3 7.909131    35.235.244.34     172.21.224.2       SSH          162 Client:
      4 7.909572    172.21.224.2      35.235.244.34      SSH          162 Server:
      5 7.910185    35.235.244.34     172.21.224.2       TCP           66 35193 →
      6 8.633082    35.235.244.34     172.21.224.2       SSH          130 Client:
      7 8.634115    172.21.224.2      35.235.244.34      SSH          194 Server:
      8 8.634316    35.235.244.34     172.21.224.2       TCP           66 35193 →
      9 8.637619    172.21.224.2      169.254.169.254    DNS           81 Standard
     10 8.637625    172.21.224.2      169.254.169.254    DNS           81 Standard
```

```
> Frame 1: 210 bytes on wire (1680 bits), 210
> Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:a
> Internet Protocol Version 4, Src: 172.21.224
> Transmission Control Protocol, Src Port: 22
> SSH Protocol
```

```
0000  42 01 ac 15 e0 01 42 01  ac 15 e0 02
0010  00 c4 da 77 40 00 40 06  bb 86 ac 15
0020  f4 22 00 16 89 79 0c b1  7a e4 de 41
0030  01 f5 a4 dc 00 00 01 01  08 0a 5b 03
0040  e7 9c 11 ed c0 7d 02 c8  cd 42 6e d4
0050  29 ba ac dc b7 82 2c 2b  25 8c eb a6
0060  7d ff ba d4 cc ab 02 e1  e2 76 1f 69
0070  e9 19 78 d9 b0 e0 1f 2a  4f 6f 45 5d
0080  88 e5 34 78 7e 54 68 9c  8d 2f fe 0d
0090  68 22 6e 7e 41 89 e1 6e  0e 65 ba fc
00a0  d9 02 f0 3d 30 b0 d5 01  b2 07 11 b9
```

sample.pcap Packets: 200 · Displayed: 200 (100.0%) Profile: Default

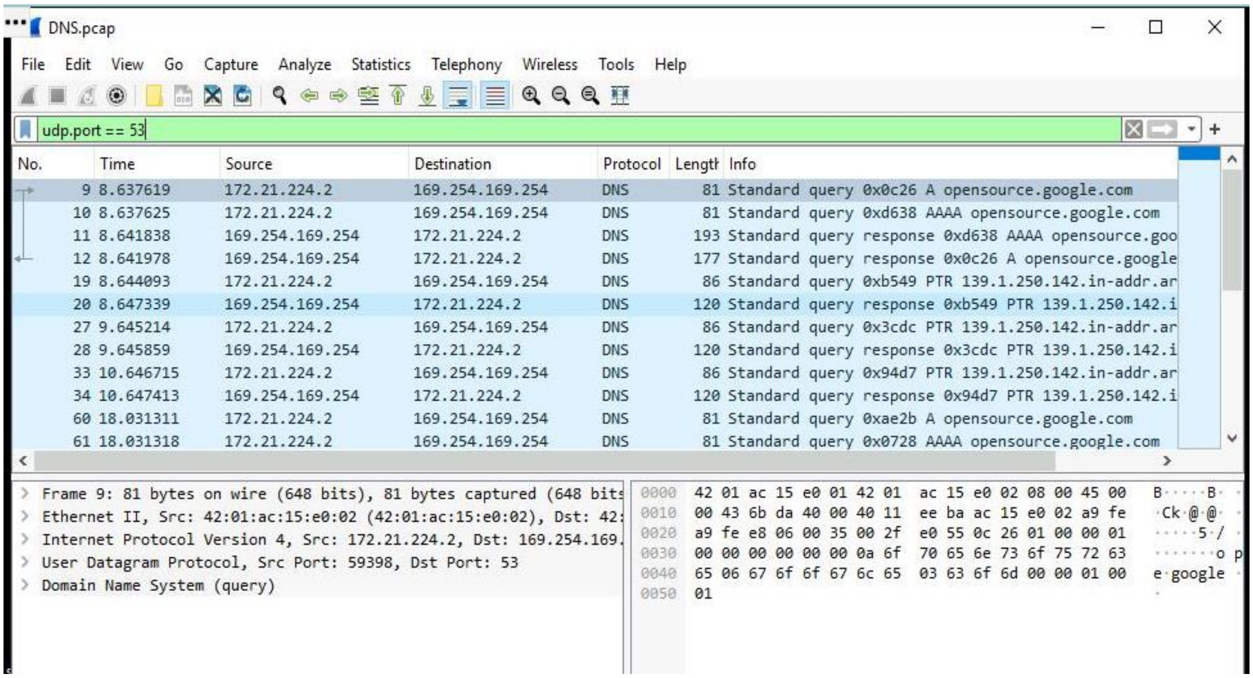
Figure 2



```
> Frame 64: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 42:01:a
> Internet Protocol Version 4, Src: 172.21.224.2, Dst: 142.250.1.139
> Transmission Control Protocol, Src Port: 49652, Dst Port: 80, Seq: 0,
```

```
0000  42 01 ac 15 e0 01 42 01  ac 15 e0 02 08 00 45 00   B·····B· ·····
0010  00 3c e4 a8 40 00 40 06  39 76 ac 15 e0 02 8e fa   ·<··@·@· 9v···
0020  01 8b c1 f4 00 50 cb 6b  93 a0 00 00 00 00 a0 02   ·····P·k ·····
0030  ff 28 1c cc 00 00 02 04  05 8c 04 02 08 0a a7 23   ·(······ ·····
0040  85 7d 00 00 00 00 01 03  03 07                     ·}······ ··
```

Wireshark filter applied to isolate HTTP traffic on TCP port 80


DNS.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
udp.port == 53
```

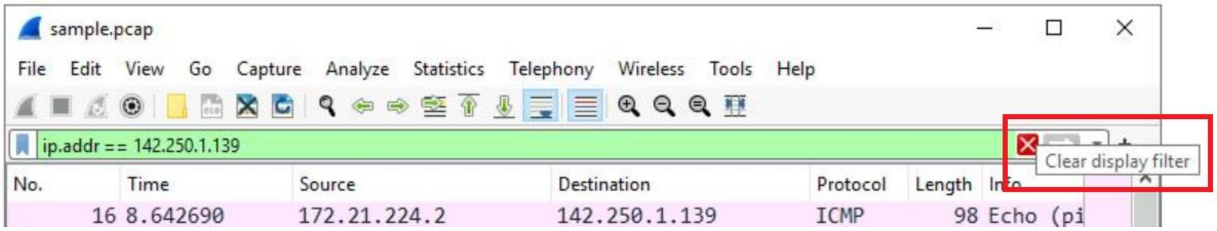
```
No.  Time       Source           Destination      Protocol Length Info
   9 8.637619   172.21.224.2     169.254.169.254  DNS       81 Standard query 0x0c26 A opensource.google.com
  10 8.637625   172.21.224.2     169.254.169.254  DNS       81 Standard query 0xd638 AAAA opensource.google.com
  11 8.641838   169.254.169.254  172.21.224.2     DNS      193 Standard query response 0xd638 AAAA opensource.goo
  12 8.641978   169.254.169.254  172.21.224.2     DNS      177 Standard query response 0x0c26 A opensource.google
  19 8.644093   172.21.224.2     169.254.169.254  DNS       86 Standard query 0xb549 PTR 139.1.250.142.in-addr.ar
  20 8.647339   169.254.169.254  172.21.224.2     DNS      120 Standard query response 0xb549 PTR 139.1.250.142.i
  27 9.645214   172.21.224.2     169.254.169.254  DNS       86 Standard query 0x3cdc PTR 139.1.250.142.in-addr.ar
  28 9.645859   169.254.169.254  172.21.224.2     DNS      120 Standard query response 0x3cdc PTR 139.1.250.142.i
  33 10.646715  172.21.224.2     169.254.169.254  DNS       86 Standard query 0x94d7 PTR 139.1.250.142.in-addr.ar
  34 10.647413  169.254.169.254  172.21.224.2     DNS      120 Standard query response 0x94d7 PTR 139.1.250.142.i
  60 18.031311  172.21.224.2     169.254.169.254  DNS       81 Standard query 0xae2b A opensource.google.com
  61 18.031318  172.21.224.2     169.254.169.254  DNS       81 Standard query 0x0728 AAAA opensource.google.com
```

```
> Frame 9: 81 bytes on wire (648 bits), 81 bytes captured (648 bits
> Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 42:
> Internet Protocol Version 4, Src: 172.21.224.2, Dst: 169.254.169.
> User Datagram Protocol, Src Port: 59398, Dst Port: 53
> Domain Name System (query)
```

```
0000  42 01 ac 15 e0 01 42 01  ac 15 e0 02 08 00 45 00   B·····B· ·
0010  00 43 6b da 40 00 40 11  ee ba ac 15 e0 02 a9 fe   ·Ck·@·@· ·
0020  a9 fe e8 06 00 35 00 2f  e0 55 0c 26 01 00 00 01   ·····5·/ ·
0030  00 00 00 00 00 00 0a 6f  70 65 6e 73 6f 75 72 63   ·······o p
0040  65 06 67 6f 6f 67 6c 65  03 63 6f 6d 00 00 01 00   e·google ·
0050  01                                                 ·
```

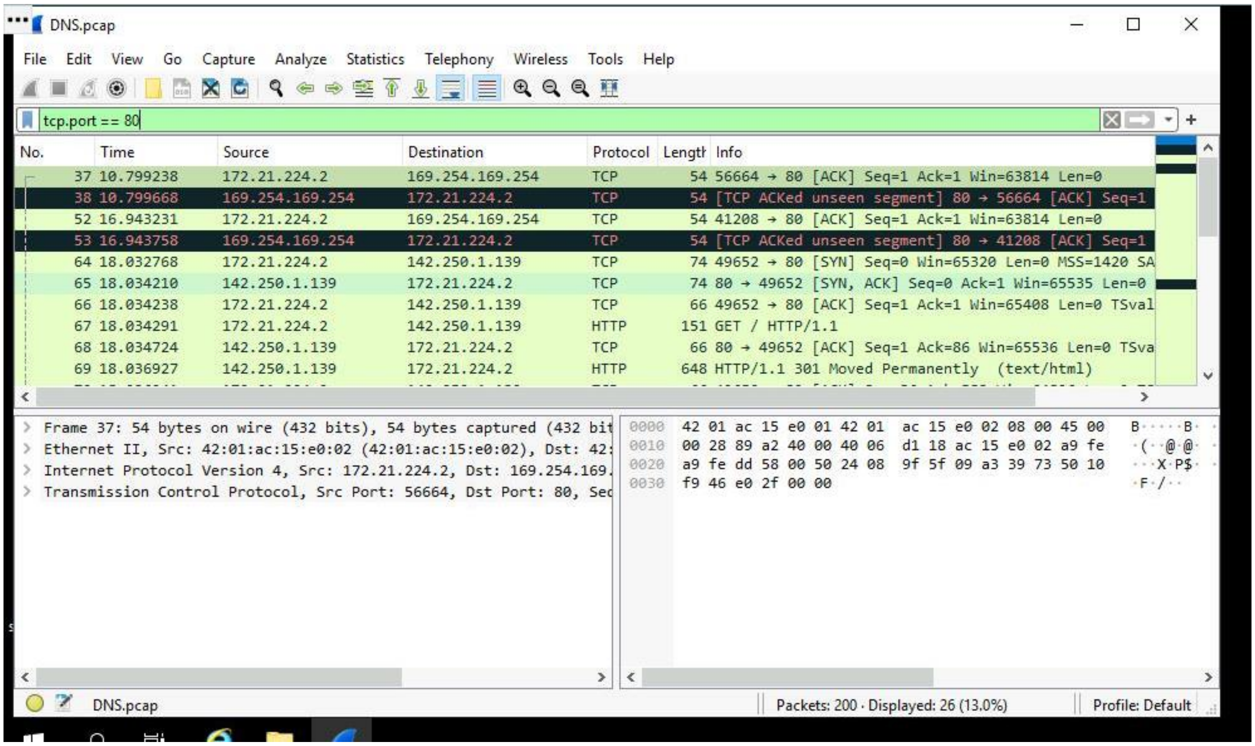
Figure 3

Caption: DNS query and response showing resolution of opensource.google.com to its IP address.


sample.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Clear display filter

```
ip.addr == 142.250.1.139
No.   Time       Source         Destination     Protocol  Length  Info
   16 8.642690   172.21.224.2   142.250.1.139   ICMP          98  Echo (pi
```

- **Figure** **4**
  Caption: ICMP echo request and reply packets captured to demonstrate basic connectivity.


DNS.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
DNS.pcap
Profile: Default

```
tcp.port == 80
No.  Time       Source           Destination      Protocol Length Info
  37 10.799238  172.21.224.2     169.254.169.254  TCP       54 56664 → 80 [ACK] Seq=1 Ack=1 Win=63814 Len=0
  38 10.799668  169.254.169.254  172.21.224.2     TCP       54 [TCP ACKed unseen segment] 80 → 56664 [ACK] Seq=1
  52 16.943231  172.21.224.2     169.254.169.254  TCP       54 41208 → 80 [ACK] Seq=1 Ack=1 Win=63814 Len=0
  53 16.943758  169.254.169.254  172.21.224.2     TCP       54 [TCP ACKed unseen segment] 80 → 41208 [ACK] Seq=1
  64 18.032768  172.21.224.2     142.250.1.139    TCP       74 49652 → 80 [SYN] Seq=0 Win=65320 Len=0 MSS=1420 SA
  65 18.034210  142.250.1.139    172.21.224.2     TCP       74 80 → 49652 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
  66 18.034238  172.21.224.2     142.250.1.139    TCP       66 49652 → 80 [ACK] Seq=1 Ack=1 Win=65408 Len=0 TSval
  67 18.034291  172.21.224.2     142.250.1.139    HTTP     151 GET / HTTP/1.1
  68 18.034724  142.250.1.139    172.21.224.2     TCP       66 80 → 49652 [ACK] Seq=1 Ack=86 Win=65536 Len=0 TSva
  69 18.036927  142.250.1.139    172.21.224.2     HTTP     648 HTTP/1.1 301 Moved Permanently  (text/html)

> Frame 37: 54 bytes on wire (432 bits), 54 bytes captured (432 bit
> Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 42:
> Internet Protocol Version 4, Src: 172.21.224.2, Dst: 169.254.169.
> Transmission Control Protocol, Src Port: 56664, Dst Port: 80, Sec

0000  42 01 ac 15 e0 01 42 01  ac 15 e0 02 08 00 45 00   B·····B·
0010  00 28 89 a2 40 00 40 06  d1 18 ac 15 e0 02 a9 fe   ·(··@·@·
0020  a9 fe dd 58 00 50 24 08  9f 5f 09 a3 39 73 50 10   ···X·P$·
0030  f9 46 e0 2f 00 00                                  ·F·/··

Packets: 200 · Displayed: 26 (13.0%)
```

- **Figure** **5**
  Caption: Detailed breakdown of a TCP packet showing Ethernet, IP, and TCP headers.

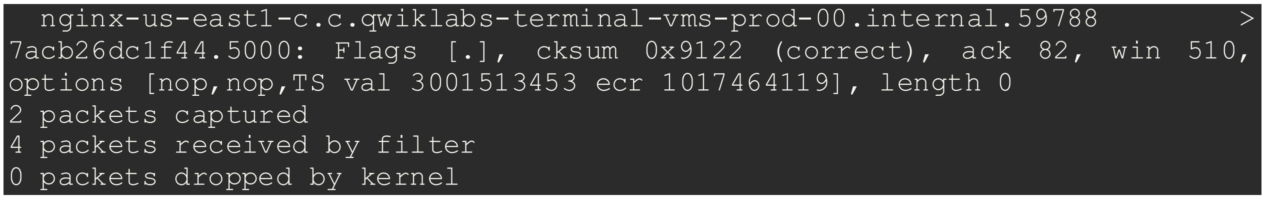
2. *Capturing Packets with Tcpdump (Linux Environment)*

- Used `ifconfig` and `tcpdump -D` to identify available network interfaces.
- Ran live captures using:
  - o `sudo tcpdump -i eth0 -v -c5` (capture 5 packets with verbose details).
  - o `sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap` (save 9 packets on port 80 to file).
- Generated HTTP traffic using `curl opensource.google.com` for testing.
- Inspected saved packet capture using:
  - o `sudo tcpdump -nn -r capture.pcap -v` (verbose packet details).
  - o `sudo tcpdump -nn -r capture.pcap -X` (hexadecimal and ASCII output for deeper inspection).
- Observed IP headers, TCP flags, and payload data for anomalies and communication patterns.

```
•  eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
•          inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
•          ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
•          RX packets 784  bytes9379957 (8.9 MiB)
•          RX errors 0  dropped 0  overruns 0  frame 0
•          TX packets 683  bytes 56880 (55.5 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
•          loop  txqueuelen 1000  (Local Loopback)
•          RX packets 400  bytes 42122 (041.1 KiB)
•          RX errors 0  dropped 0  overruns 0  frame 0
•          TX packets 400  bytes 42122 (041.1 KiB)
•          TX errors 0  dropped 0 overruns 0  carrier 0 collisions 0
```

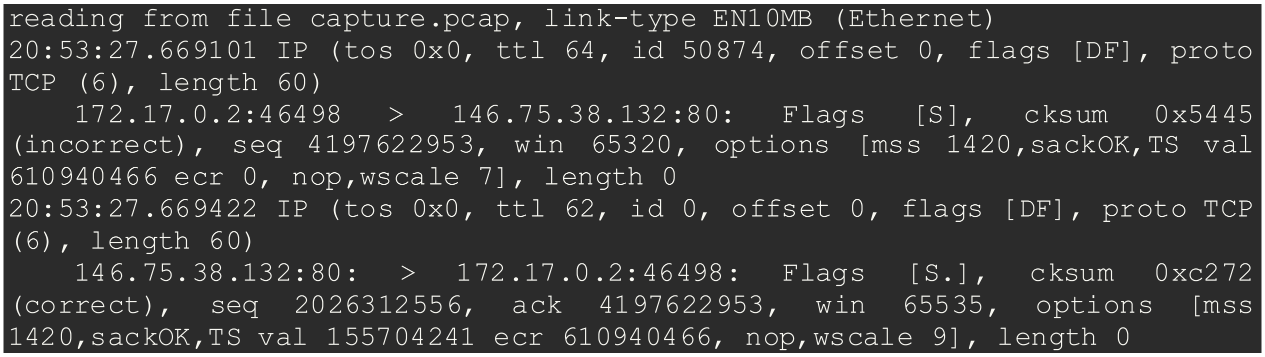
Fig. 6
tcpdump command-line identifying network interface

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
10:57:33.427749 IP (tos 0x0, ttl 64, id 35057, offset 0, flags [DF], protot
TCP (6), length 134)
  7acb26dc1f44.5000    >    nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-
00.internal.59788:  Flags  [P.],  cksum  0x5851  (incorrect  >  0x30d3),  seq
1080713945:1080714027,  ack  62760789,  win  501,  options  [nop,nop,TS  val
1017464119 ecr 3001513453], length 82
10:57:33.427954 IP (tos 0x0, ttl 64, id 21812, offset 0, flags [DF], protot
TCP (6), length 52)
```



```
  nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-00.internal.59788        >
7acb26dc1f44.5000: Flags [.], cksum 0x9122 (correct), ack 82, win 510,
options [nop,nop,TS val 3001513453 ecr 1017464119], length 0
2 packets captured
4 packets received by filter
0 packets dropped by kernel
```

Fig. 7. **Inspect the network traffic of a network interface with tcpdump**



```
reading from file capture.pcap, link-type EN10MB (Ethernet)
20:53:27.669101 IP (tos 0x0, ttl 64, id 50874, offset 0, flags [DF], proto
TCP (6), length 60)
    172.17.0.2:46498   >   146.75.38.132:80:   Flags   [S],   cksum   0x5445
(incorrect), seq 4197622953, win 65320, options [mss 1420,sackOK,TS val
610940466 ecr 0, nop,wscale 7], length 0
20:53:27.669422 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto TCP
(6), length 60)
    146.75.38.132:80:  >  172.17.0.2:46498:  Flags  [S.],  cksum  0xc272
(correct),  seq  2026312556,  ack  4197622953,  win  65535,  options  [mss
1420,sackOK,TS val 155704241 ecr 610940466, nop,wscale 9], length 0
```

Fig. 8. **Filter the captured packet data.**

Outcome & Skills Gained

This project strengthened my ability to:

- Capture and interpret live and stored network traffic
- Use filters efficiently to investigate relevant data
- Differentiate between protocols (DNS, TCP, ICMP, etc.)
- Apply packet analysis in the context of **network security monitoring**