GOAL: Capture and analyze live network traffic from a Linux VM using tcpdump to familiarize myself with it..

First, I will capture network packet data. I will then use tcpdump to filter live network traffic. Third, I will capture network traffic using tcpdump. Finally, I will filter the captured packet data.

```
analyst@b920e4e62e60:~$ sudo ifconfig
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 620  bytes 13697560 (13.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 328  bytes 31842 (31.0 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 60  bytes 8601 (8.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 60  bytes 8601 (8.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

I first use ifconfig with admin privileges (sudo) to identify the interfaces that are available. I will be using eth0 as the interface to capture the network packet data.

```
analyst@b920e4e62e60:~$ sudo tcpdump -D
1.eth0 [Up, Running]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.nflog (Linux netfilter log (NFLOG) interface)
5.nfqueue (Linux netfilter queue (NFQUEUE) interface)
```

Here, I use tcpdump to identify the interface options available for packet capture. I can also see which ones are available and can be useful if the system does not support ifconfig command.

```
analyst@b920e4e62e60:~$ sudo tcpdump -i eth0 -v -c5
```

I run this to capture from eth0 interface with verbiage (detailed packet data) and only capturing 5 packets of data.

The output

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
06:26:56.540543 IP (tos 0x0, ttl 64, id 39774, offset 0, flags [DF], proto TCP (6), length 114)
    b920e4e62e60.5000 > nginx-us-east1-b.c.qwiklabs-terminal-vms-prod-00.internal.59542: Flags [P.], ck
sum 0x588c (incorrect -> 0x3383), seq 3741501217:3741501279, ack 4108516597, win 501, options [nop,nop,
TS val 662505640 ecr 2096584087], length 62
06:26:56.540852 IP (tos 0x0, ttl 63, id 54230, offset 0, flags [DF], proto TCP (6), length 52)
    nginx-us-east1-b.c.qwiklabs-terminal-vms-prod-00.internal.59542 > b920e4e62e60.5000: Flags [.], cks
um 0x8497 (correct), ack 62, win 507, options [nop,nop,TS val 2096584332 ecr 662505640], length 0
06:26:56.551128 IP (tos 0x0, ttl 64, id 39775, offset 0, flags [DF], proto TCP (6), length 146)
    b920e4e62e60.5000 > nginx-us-east1-b.c.qwiklabs-terminal-vms-prod-00.internal.59542: Flags [P.], ck
sum 0x58ac (incorrect -> 0xf437), seq 62:156, ack 1, win 501, options [nop,nop,TS val 662505651 ecr 209
6584332], length 94
06:26:56.551296 IP (tos 0x0, ttl 63, id 54231, offset 0, flags [DF], proto TCP (6), length 52)
    nginx-us-east1-b.c.qwiklabs-terminal-vms-prod-00.internal.59542 > b920e4e62e60.5000: Flags [.], cks
um 0x8424 (correct), ack 156, win 507, options [nop,nop,TS val 2096584342 ecr 662505651], length 0
06:26:56.592469 IP (tos 0x0, ttl 64, id 44887, offset 0, flags [DF], proto UDP (17), length 69)
    b920e4e62e60.53320 > metadata.google.internal.domain: 18906+ PTR? 2.0.18.172.in-addr.arpa. (41)
5 packets captured
8 packets received by filter
0 packets dropped by kernel
```

It is listening to the eth0 interface and it provided with information on the link type (EN10MB Ethernet) and the capture size (262144 bytes). A single verso option shows packet fields like TOS TTL (Time to Live), offset, flags, internal protocol type and the length of the outer IP packet.

```
analyst@b920e4e62e60:~$ sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap &
```

This captures 9 packets of data from port 80 which is the HTTP port and writes the captured data in a file. The & is to run the Bash shell command in the background..

```
analyst@b920e4e62e60:~$ curl opensource.google.com
```

I ran curl to generate some traffic with HTTP under port 80.

```
analyst@b920e4e62e60:~$ ls -l capture.pcap
-rw-r--r-- 1 root root 1445 Nov 28 06:53 capture.pcap
```

I can check that the packet data was captured.

```
analyst@b920e4e62e60:~$ sudo tcpdump -nn -r capture.pcap -v
```

I use tcpdump here to filter the packet header from the capture file. -nn is used here to disable port and protocol name lookup which makes it so tcpdump does not perform name lookups of either any IP addresses or ports. This can alert threat actors when a name lookup is processed.