Activity overview

As a security analyst, it’s important to know how to capture and filter network traffic in a Linux environment. You’ll also need to know the basic concepts associated with network interfaces.

In this lab activity, you’ll perform tasks associated with using tcpdump to capture network traffic. You’ll capture the data in a packet capture (p-cap) file and then examine the contents of the captured packet data to focus on specific types of traffic.

Let’s capture network traffic!

Scenario

You’re a network analyst who needs to use `tcpdump` to capture and analyze live network traffic from a Linux virtual machine.

The lab starts with your user account, called `analyst`, already logged in to a Linux terminal.

Your Linux user's home directory contains a sample packet capture file that you will use at the end of the lab to answer a few questions about the network traffic that it contains.

Here’s how you’ll do this: **First**, you’ll identify network interfaces to capture network packet data. **Second**, you’ll use `tcpdump` to filter live network traffic. **Third**, you’ll capture network traffic using `tcpdump`. **Finally**, you’ll filter the captured packet data.

Disclaimer: For optimal performance and compatibility, it is recommended to use either **Google Chrome** or **Mozilla Firefox** browsers while accessing the labs.

Start your lab

Before you begin, you can review the instructions for using the Qwiklabs platform under the **Resources** tab in Coursera.

If you haven't already done so, click **Start Lab**. This brings up the terminal so that you can begin completing the tasks!

When you have completed all the tasks, refer to the **End your Lab** section that follows the tasks for information on how to end your lab.

Task 1. Identify network interfaces

In this task, you must identify the network interfaces that can be used to capture network packet data.

1. Use `ifconfig` to identify the interfaces that are available:

```
sudo ifconfig
```

Copied!
This command returns output similar to the following:

```
eth0: flags=4163  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  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
```
The Ethernet network interface is identified by the entry with the `eth` prefix.

So, in this lab, you'll use `eth0` as the interface that you will capture network packet data from in the following tasks.

2. Use `tcpdump` to identify the interface options available for packet capture:

```
sudo tcpdump -D
```
Copied!

This command will also allow you to identify which network interfaces are available. This may be useful on systems that do not include the `ifconfig` command.

Click **Check my progress** to verify that you have completed this task correctly.

You have completed this task and used ifconfig and tcpdump to list the network interfaces in this machine.

Identify network interfaces

You have completed this task and used ifconfig and tcpdump to list the network interfaces in this machine.

```
nalyst@c0b9c7449e82:~$ 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 701  bytes 13728622 (13.0 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 323  bytes 31620 (30.8 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 94  bytes 11841 (11.5 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 94  bytes 11841 (11.5 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

analyst@c0b9c7449e82:~$ sudo tcpdump -d
(000) ret      #262144
analyst@c0b9c7449e82:~$ 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)
```

Task 2. Inspect the network traffic of a network interface with tcpdump

In this task, you must use `tcpdump` to filter live network packet traffic on an interface.

- Filter live network packet data from the `eth0` interface with `tcpdump`:

```
sudo tcpdump -i eth0 -v -c5
```

Copied!

This command will run `tcpdump` with the following options:

- `-i eth0`: Capture data specifically from the `eth0` interface.
- `-v`: Display detailed packet data.
- `-c5`: Capture 5 packets of data.

Now, let's take a detailed look at the packet information that this command has returned.

Some of your packet traffic data will be similar to the following:

```
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
```

The specific packet data in your lab may be in a different order and may even be for entirely different types of network traffic. The specific details, such as system names, ports, and checksums, will definitely be different. You can run this command again to get different snapshots to outline how data changes between packets.

Exploring network packet details

In this example, you'll identify some of the properties that `tcpdump` outputs for the packet capture data you've just seen.

1. In the example data at the start of the packet output, `tcpdump` reported that it was listening on the `eth0` interface, and it provided information on the link type and the capture size in bytes:

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size
262144 bytes
```

2. On the next line, the first field is the packet's timestamp, followed by the protocol type, IP:

```
22:24:18.910372 IP
```

3. The verbose option, `-v`, has provided more details about the IP packet fields, such as TOS, TTL, offset, flags, internal protocol type (in this case, TCP (6)), and the length of the outer IP packet in bytes:

```
(tos 0x0, ttl 64, id 5802, offset 0, flags [DF], proto TCP (6), length
134)
```

The specific details about these fields are beyond the scope of this lab. But you should know that these are properties that relate to the IP network packet.

4. In the next section, the data shows the systems that are communicating with each other:

```
7acb26dc1f44.5000 > nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-
00.internal.59788:
```

By default, `tcpdump` will convert IP addresses into names, as in the screenshot. The name of your Linux virtual machine, also included in the command prompt, appears here as the source for one packet and the destination for the second packet. In your live data, the name will be a different set of letters and numbers.

The direction of the arrow (>) indicates the direction of the traffic flow in this packet. Each system name includes a suffix with the port number (.5000 in the screenshot), which is used by the source and the destination systems for this packet.

5. The remaining data filters the header data for the inner TCP packet:

```
Flags [P.], cksum 0x5851 (incorrect > 0x30d3), seq 1080713945:1080714027,
ack 62760789, win 501, options [nop,nop,TS val 1017464119 ecr 3001513453],
length 82
```

The flags field identifies TCP flags. In this case, the P represents the push flag and the period indicates it's an ACK flag. This means the packet is pushing out data.

The next field is the TCP checksum value, which is used for detecting errors in the data.

This section also includes the sequence and acknowledgment numbers, the window size, and the length of the inner TCP packet in bytes.

Click **Check my progress** to verify that you have completed this task correctly.

You have completed this task and used tcpdump to filter live network traffic.

Inspect network traffic with tcpdump

You have completed this task and used tcpdump to filter live network traffic.

```
nalyst@c0b9c7449e82:~$ sudo tcpdump -i eth0 -v -c5

tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

19:07:09.139504 IP (tos 0x0, ttl 64, id 24609, offset 0, flags [DF], proto TCP (6), length 112)

   c0b9c7449e82.5000 > nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-
00.internal.58646: Flags [P.], cksum 0x588d (incorrect -> 0xb648), seq
702848947:702849007, ack 3375769933, win 501, options [nop,nop,TS val 2540888704 ecr
1803454428], length 60

19:07:09.139820 IP (tos 0x0, ttl 63, id 54120, offset 0, flags [DF], proto TCP (6), length 52)

   nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-00.internal.58646 >
c0b9c7449e82.5000: Flags [.], cksum 0xf975 (correct), ack 60, win 507, options
[nop,nop,TS val 1803454649 ecr 2540888704], length 0

19:07:09.150249 IP (tos 0x0, ttl 64, id 24610, offset 0, flags [DF], proto TCP (6), length 147)

   c0b9c7449e82.5000 > nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-
00.internal.58646: Flags [P.], cksum 0x58b0 (incorrect -> 0xd525), seq 60:155, ack 1, win
501, options [nop,nop,TS val 2540888715 ecr 1803454649], length 95

19:07:09.150506 IP (tos 0x0, ttl 63, id 54121, offset 0, flags [DF], proto TCP (6), length 52)

   nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-00.internal.58646 >
c0b9c7449e82.5000: Flags [.], cksum 0xf900 (correct), ack 155, win 507, options
[nop,nop,TS val 1803454660 ecr 2540888715], length 0

19:07:09.164663 IP (tos 0x0, ttl 64, id 39340, offset 0, flags [DF], proto UDP (17), length 69)

   c0b9c7449e82.43276 > metadata.google.internal.domain: 8738+ PTR? 2.0.21.172.in-
addr.arpa. (41)

5 packets captured

10 packets received by filter

0 packets dropped by kernel
```

```
analyst@c0b9c7449e82:~$
```

Task 3. Capture network traffic with tcpdump

In this task, you will use `tcpdump` to save the captured network data to a packet capture file.

In the previous command, you used `tcpdump` to stream all network traffic. Here, you will use a filter and other `tcpdump` configuration options to save a small sample that contains only web (TCP port 80) network packet data.

1. Capture packet data into a file called `capture.pcap`:

```
sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap &
```

Copied!

You must press the **ENTER** key to get your command prompt back after running this command.

This command will run `tcpdump` in the background with the following options:

- `-i eth0`: Capture data from the `eth0` interface.
- `-nn`: Do not attempt to resolve IP addresses or ports to names.This is best practice from a security perspective, as the lookup data may not be valid. It also prevents malicious actors from being alerted to an investigation.
- `-c9`: Capture 9 packets of data and then exit.
- `port 80`: Filter only port 80 traffic. This is the default HTTP port.
- `-w capture.pcap`: Save the captured data to the named file.
- `&`: This is an instruction to the Bash shell to run the command in the background.

This command runs in the background, but some output text will appear in your terminal. The text will not affect the commands when you follow the steps for the rest of the lab.

2. Use `curl` to generate some HTTP (port 80) traffic:

```
curl opensource.google.com
```

Copied!
When the `curl` command is used like this to open a website, it generates some HTTP (TCP port 80) traffic that can be captured.

3. Verify that packet data has been captured:

```
ls -l capture.pcap
```

Copied!
Note: *The "Done" in the output indicates that the packet was captured.*
Click **Check my progress** to verify that you have completed this task correctly.

You have completed this task. You used the tcpdump command to capture network data to the file /home/analyst/capture.pcap.

Capture network traffic with tcpdump

You have completed this task. You used the tcpdump command to capture network data to the file /home/analys

```
analyst@c0b9c7449e82:~$ sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap &

[1] 12789

analyst@c0b9c7449e82:~$ tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes


analyst@c0b9c7449e82:~$ curl opensource.google.com

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">

<TITLE>301 Moved</TITLE></HEAD><BODY>

<H1>301 Moved</H1>

The document has moved

<A HREF="https://opensource.google/">here</A>.

</BODY></HTML>

9 packets captured

12 packets received by filter

0 packets dropped by kernel
```

```
[1]+  Done                 sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap

analyst@c0b9c7449e82:~$ ls -al capture.pcap

-rw-r--r-- 1 root root 1401 May 10 19:11 capture.pcap

analyst@c0b9c7449e82:~$
```

Task 4. Filter the captured packet data

In this task, use `tcpdump` to filter data from the packet capture file you saved previously.

1. Use the `tcpdump` command to filter the packet header data from the `capture.pcap` capture file:

```
sudo tcpdump -nn -r capture.pcap -v
```

Copied!

This command will run `tcpdump` with the following options:

- `-nn`: Disable port and protocol name lookup.
- `-r`: Read capture data from the named file.
- `-v`: Display detailed packet data.

You must specify the `-nn` switch again here, as you want to make sure `tcpdump` does not perform name lookups of either IP addresses or ports, since this can alert threat actors.

This returns output data similar to the following:

```
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
```

As in the previous example, you can see the IP packet information along with information about the data that the packet contains.

2. Use the `tcpdump` command to filter the extended packet data from the `capture.pcap` capture file:

```
sudo tcpdump -nn -r capture.pcap -X
```

Copied!

This command will run `tcpdump` with the following options:

- `-nn`: Disable port and protocol name lookup.
- `-r`: Read capture data from the named file.
- `-X`: Display the hexadecimal and ASCII output format packet data. Security analysts can analyze hexadecimal and ASCII output to detect patterns or anomalies during malware analysis or forensic analysis.

***Note:** Hexadecimal, also known as hex or base 16, uses 16 symbols to represent values, including the digits 0-9 and letters A, B, C, D, E, and F. American Standard Code for Information Interchange (ASCII) is a character encoding standard that uses a set of characters to represent text in digital form.*

Click **Check my progress** to verify that you have completed this task correctly.

You have completed this task and used tcpdump to view saved data in a saved packet capture file.

Filter the captured packet data

You have completed this task and used tcpdump to view saved data in a saved packet capture file.

```
9c7449e82:~$ sudo tcpdump -nn -r capture.pcap -v

reading from file capture.pcap, link-type EN10MB (Ethernet)

19:11:14.549177 IP (tos 0x0, ttl 64, id 15834, offset 0, flags [DF], proto TCP (6), length 60)
```

```
   172.17.0.2.46810 > 173.194.217.101.80: Flags [S], cksum 0x336a (incorrect -> 0xcbd2), seq 2609622726, win 65320, options [mss 1420,sackOK,TS val 1536552158 ecr 0,nop,wscale 7], length 0

19:11:14.550296 IP (tos 0x60, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length 60)

   173.194.217.101.80 > 172.17.0.2.46810: Flags [S.], cksum 0x4a25 (correct), seq 1516992576, ack 2609622727, win 65535, options [mss 1420,sackOK,TS val 2679836253 ecr 1536552158,nop,wscale 8], length 0

19:11:14.550318 IP (tos 0x0, ttl 64, id 15835, offset 0, flags [DF], proto TCP (6), length 52)

   172.17.0.2.46810 > 173.194.217.101.80: Flags [.], cksum 0x3362 (incorrect -> 0x76ca), ack 1, win 511, options [nop,nop,TS val 1536552159 ecr 2679836253], length 0

19:11:14.550403 IP (tos 0x0, ttl 64, id 15836, offset 0, flags [DF], proto TCP (6), length 137)

   172.17.0.2.46810 > 173.194.217.101.80: Flags [P.], cksum 0x33b7 (incorrect -> 0xe57d), seq 1:86, ack 1, win 511, options [nop,nop,TS val 1536552159 ecr 2679836253], length 85: HTTP, length: 85

      GET / HTTP/1.1

      Host: opensource.google.com

      User-Agent: curl/7.64.0

      Accept: */*


19:11:14.550753 IP (tos 0x60, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length 52)

   173.194.217.101.80 > 172.17.0.2.46810: Flags [.], cksum 0x7773 (correct), ack 86, win 256, options [nop,nop,TS val 2679836254 ecr 1536552159], length 0

19:11:14.553711 IP (tos 0x80, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length 590)

   173.194.217.101.80 > 172.17.0.2.46810: Flags [P.], cksum 0x5c6f (correct), seq 1:539, ack 86, win 256, options [nop,nop,TS val 2679836257 ecr 1536552159], length 538: HTTP, length: 538

      HTTP/1.1 301 Moved Permanently

      Location: https://opensource.google/

      Content-Type: text/html; charset=UTF-8
```

```
        X-Content-Type-Options: nosniff
        Date: Fri, 10 May 2024 19:11:14 GMT
        Expires: Fri, 10 May 2024 19:41:14 GMT
        Cache-Control: public, max-age=1800
        Server: sffe
        Content-Length: 223
        X-XSS-Protection: 0

        <HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
        <TITLE>301 Moved</TITLE></HEAD><BODY>
        <H1>301 Moved</H1>
        The document has moved
        <A HREF="https://opensource.google/">here</A>.
        </BODY></HTML>
19:11:14.553728 IP (tos 0x0, ttl 64, id 15837, offset 0, flags [DF], proto TCP (6), length 52)
    172.17.0.2.46810 > 173.194.217.101.80: Flags [.], cksum 0x3362 (incorrect -> 0x7458), ack 539, win 507, options [nop,nop,TS val 1536552162 ecr 2679836257], length 0
19:11:14.555305 IP (tos 0x0, ttl 64, id 15838, offset 0, flags [DF], proto TCP (6), length 52)
    172.17.0.2.46810 > 173.194.217.101.80: Flags [F.], cksum 0x3362 (incorrect -> 0x7455), seq 86, ack 539, win 507, options [nop,nop,TS val 1536552164 ecr 2679836257], length 0
19:11:14.555664 IP (tos 0x80, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length 52)
    173.194.217.101.80 > 172.17.0.2.46810: Flags [F.], cksum 0x754d (correct), seq 539, ack 87, win 256, options [nop,nop,TS val 2679836259 ecr 1536552164], length 0
analyst@c0b9c7449e82:~$ sudo tcpdump -nn -r capture.pcap -X
reading from file capture.pcap, link-type EN10MB (Ethernet)
19:11:14.549177 IP 172.17.0.2.46810 > 173.194.217.101.80: Flags [S], seq 2609622726, win 65320, options [mss 1420,sackOK,TS val 1536552158 ecr 0,nop,wscale 7], length 0
```

```
    0x0000:  4500 003c 3dda 4000 4006 c9a6 ac11 0002  E..<=.@.@.......
    0x0010:  adc2 d965 b6da 0050 9b8b aec6 0000 0000  ...e...P........
    0x0020:  a002 ff28 336a 0000 0204 058c 0402 080a  ...(3j..........
    0x0030:  5b95 ecde 0000 0000 0103 0307            [...........
19:11:14.550296 IP 173.194.217.101.80 > 172.17.0.2.46810: Flags [S.], seq 1516992576,
ack 2609622727, win 65535, options [mss 1420,sackOK,TS val 2679836253 ecr
1536552158,nop,wscale 8], length 0
    0x0000:  4560 003c 0000 4000 7e06 c920 adc2 d965  E`.<..@.~......e
    0x0010:  ac11 0002 0050 b6da 5a6b 7840 9b8b aec7  .....P..Zkx@....
    0x0020:  a012 ffff 4a25 0000 0204 058c 0402 080a  ....J%..........
    0x0030:  9fbb 0e5d 5b95 ecde 0103 0308            ...][.......
19:11:14.550318 IP 172.17.0.2.46810 > 173.194.217.101.80: Flags [.], ack 1, win 511,
options [nop,nop,TS val 1536552159 ecr 2679836253], length 0
    0x0000:  4500 0034 3ddb 4000 4006 c9ad ac11 0002  E..4=.@.@.......
    0x0010:  adc2 d965 b6da 0050 9b8b aec7 5a6b 7841  ...e...P....ZkxA
    0x0020:  8010 01ff 3362 0000 0101 080a 5b95 ecdf  ....3b......[...
    0x0030:  9fbb 0e5d                                ...]
19:11:14.550403 IP 172.17.0.2.46810 > 173.194.217.101.80: Flags [P.], seq 1:86, ack 1, win
511, options [nop,nop,TS val 1536552159 ecr 2679836253], length 85: HTTP: GET /
HTTP/1.1
    0x0000:  4500 0089 3ddc 4000 4006 c957 ac11 0002  E...=.@.@..W....
    0x0010:  adc2 d965 b6da 0050 9b8b aec7 5a6b 7841  ...e...P....ZkxA
    0x0020:  8018 01ff 33b7 0000 0101 080a 5b95 ecdf  ....3.......[...
    0x0030:  9fbb 0e5d 4745 5420 2f20 4854 5450 2f31  ...]GET./.HTTP/1
    0x0040:  2e31 0d0a 486f 7374 3a20 6f70 656e 736f  .1..Host:.openso
    0x0050:  7572 6365 2e67 6f6f 676c 652e 636f 6d0d  urce.google.com.
    0x0060:  0a55 7365 722d 4167 656e 743a 2063 7572  .User-Agent:.cur
```

```
        0x0070:  6c2f 372e 3634 2e30 0d0a 4163 6365 7074  l/7.64.0..Accept
        0x0080:  3a20 2a2f 2a0d 0a0d 0a                   :.*/*....
19:11:14.550753 IP 173.194.217.101.80 > 172.17.0.2.46810: Flags [.], ack 86, win 256, options [nop,nop,TS val 2679836254 ecr 1536552159], length 0
        0x0000:  4560 0034 0000 4000 7e06 c928 adc2 d965  E`.4..@.~..(...e
        0x0010:  ac11 0002 0050 b6da 5a6b 7841 9b8b af1c  .....P..ZkxA....
        0x0020:  8010 0100 7773 0000 0101 080a 9fbb 0e5e  ....ws.........^
        0x0030:  5b95 ecdf                                [...
19:11:14.553711 IP 173.194.217.101.80 > 172.17.0.2.46810: Flags [P.], seq 1:539, ack 86, win 256, options [nop,nop,TS val 2679836257 ecr 1536552159], length 538: HTTP: HTTP/1.1 301 Moved Permanently
        0x0000:  4580 024e 0000 4000 7e06 c6ee adc2 d965  E..N..@.~......e
        0x0010:  ac11 0002 0050 b6da 5a6b 7841 9b8b af1c  .....P..ZkxA....
        0x0020:  8018 0100 5c6f 0000 0101 080a 9fbb 0e61  ....\o.........a
        0x0030:  5b95 ecdf 4854 5450 2f31 2e31 2033 3031  [...HTTP/1.1.301
        0x0040:  204d 6f76 6564 2050 6572 6d61 6e65 6e74  .Moved.Permanent
        0x0050:  6c79 0d0a 4c6f 6361 7469 6f6e 3a20 6874  ly..Location:.ht
        0x0060:  7470 733a 2f2f 6f70 656e 736f 7572 6365  tps://opensource
        0x0070:  2e67 6f6f 676c 652f 0d0a 436f 6e74 656e  .google/..Conten
        0x0080:  742d 5479 7065 3a20 7465 7874 2f68 746d  t-Type:.text/htm
        0x0090:  6c3b 2063 6861 7273 6574 3d55 5446 2d38  l;.charset=UTF-8
        0x00a0:  0d0a 582d 436f 6e74 656e 742d 5479 7065  ..X-Content-Type
        0x00b0:  2d4f 7074 696f 6e73 3a20 6e6f 736e 6966  -Options:.nosnif
        0x00c0:  660d 0a44 6174 653a 2046 7269 2c20 3130  f..Date:.Fri,.10
        0x00d0:  204d 6179 2032 3032 3420 3139 3a31 313a  .May.2024.19:11:
        0x00e0:  3134 2047 4d54 0d0a 4578 7069 7265 733a  14.GMT..Expires:
        0x00f0:  2046 7269 2c20 3130 204d 6179 2032 3032  .Fri,.10.May.202
        0x0100:  3420 3139 3a34 313a 3134 2047 4d54 0d0a  4.19:41:14.GMT..
        0x0110:  4361 6368 652d 436f 6e74 726f 6c3a 2070  Cache-Control:.p
        0x0120:  7562 6c69 632c 206d 6178 2d61 6765 3d31  ublic,.max-age=1
        0x0130:  3830 300d 0a53 6572 7665 723a 2073 6666  800..Server:.sff
        0x0140:  650d 0a43 6f6e 7465 6e74 2d4c 656e 6774  e..Content-Lengt
        0x0150:  683a 2032 3233 0d0a 582d 5853 532d 5072  h:.223..X-XSS-Pr
        0x0160:  6f74 6563 7469 6f6e 3a20 300d 0a0d 0a3c  otection:.0....<
        0x0170:  4854 4d4c 3e3c 4845 4144 3e3c 6d65 7461  HTML><HEAD><meta
        0x0180:  2068 7474 702d 6571 7569 763d 2263 6f6e  .http-equiv="con
        0x0190:  7465 6e74 2d74 7970 6522 2063 6f6e 7465  tent-type".conte
        0x01a0:  6e74 3d22 7465 7874 2f68 746d 6c3b 6368  nt="text/html;ch
        0x01b0:  6172 7365 743d 7574 662d 3822 3e0a 3c54  arset=utf-8">.<T
        0x01c0:  4954 4c45 3e33 3031 204d 6f76 6564 3c2f  ITLE>301.Moved</
        0x01d0:  5449 544c 453e 3c2f 4845 4144 3e3c 424f  TITLE></HEAD><BO
        0x01e0:  4459 3e0a 3c48 313e 3330 3120 4d6f 7665  DY>.<H1>301.Move
        0x01f0:  643c 2f48 313e 0a54 6865 2064 6f63 756d  d</H1>.The.docum
        0x0200:  656e 7420 6861 7320 6d6f 7665 640a 3c41  ent.has.moved.<A
        0x0210:  2048 5245 463d 2268 7474 7073 3a2f 2f6f  .HREF="https://o
        0x0220:  7065 6e73 6f75 7263 652e 676f 6f67 6c65  pensource.google
        0x0230:  2f22 3e68 6572 653c 2f41 3e2e 0d0a 3c2f  /">here</A>...</
        0x0240:  424f 4459 3e3c 2f48 544d 4c3e 0d0a       BODY></HTML>..
19:11:14.553728 IP 172.17.0.2.46810 > 173.194.217.101.80: Flags [.], ack 539, win 507, options [nop,nop,TS val 1536552162 ecr 2679836257], length 0
        0x0000:  4500 0034 3ddd 4000 4006 c9ab ac11 0002  E..4=.@.@.......
        0x0010:  adc2 d965 b6da 0050 9b8b af1c 5a6b 7a5b  ...e...P....Zkz[
        0x0020:  8010 01fb 3362 0000 0101 080a 5b95 ece2  ....3b......[...
```

```
    0x0030:  9fbb 0e61                             ...a

19:11:14.555305 IP 172.17.0.2.46810 > 173.194.217.101.80: Flags [F.], seq 86, ack 539, win 507, options [nop,nop,TS val 1536552164 ecr 2679836257], length 0

    0x0000:  4500 0034 3dde 4000 4006 c9aa ac11 0002  E..4=.@.@.......

    0x0010:  adc2 d965 b6da 0050 9b8b af1c 5a6b 7a5b  ...e...P....Zkz[

    0x0020:  8011 01fb 3362 0000 0101 080a 5b95 ece4  ....3b......[...

    0x0030:  9fbb 0e61                             ...a

19:11:14.555664 IP 173.194.217.101.80 > 172.17.0.2.46810: Flags [F.], seq 539, ack 87, win 256, options [nop,nop,TS val 2679836259 ecr 1536552164], length 0

    0x0000:  4580 0034 0000 4000 7e06 c908 adc2 d965  E..4..@.~......e

    0x0010:  ac11 0002 0050 b6da 5a6b 7a5b 9b8b af1d  .....P..Zkz[....

    0x0020:  8011 0100 754d 0000 0101 080a 9fbb 0e63  ....uM.........c

    0x0030:  5b95 ece4                             [...

analyst@c0b9c7449e82:~$
```

Test your understanding

To test your ability to capture and view network data, answer the multiple-choice questions.

What command would you use to capture 3 packets on any interface with the verbose option?
checksudo tcpdump -c3 -i any -v
sudo tcpdump -s3 -i all -v
sudo tcpdump -N2 -i any -v
sudo tcpdump -n3 -i any -v

What does the -i option indicate?
The number of packets to capture
checkThe network interface to monitor
Incremental monitoring mode

Capture incoming packets only

What type of information does the -v option include?

Packets including the letter `V`

Version information

checkVerbose information

Virtual packets

What tcpdump command can you use to identify the interfaces that are available to perform a packet capture on?

sudo tcpdump

checksudo tcpdump -D

sudo ls

sudo capture p.cap