Activity: Capture your first packet

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.

Task 1. Identify network interfaces

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

```
analyst@457a0067943c:~$ 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 688  bytes 13739741 (13.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 345  bytes 33041 (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
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 90  bytes 11633 (11.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 90  bytes 11633 (11.3 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.

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

```
analyst@457a0067943c:~$ 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)
```

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.

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

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

```
analyst@457a0067943c:~$ sudo tcpdump -i eth0 -v -c5
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:53:34.191104 IP (tos 0x0, ttl 64, id 50428, offset 0, flags [DF], proto TCP (6), len
gth 114)
    457a0067943c.5000 > nginx-us-east1-b.c.qwiklabs-terminal-vms-prod-00.internal.46630
: Flags [P.], cksum 0x588c (incorrect -> 0xf406), seq 351226120:351226182, ack 12438016
67, win 492, options [nop,nop,TS val 2619955275 ecr 2136146523], length 62
09:53:34.191443 IP (tos 0x0, ttl 63, id 55120, offset 0, flags [DF], proto TCP (6), len
gth 52)
    nginx-us-east1-b.c.qwiklabs-terminal-vms-prod-00.internal.46630 > 457a0067943c.5000
: Flags [.], cksum 0x4bff (correct), ack 62, win 507, options [nop,nop,TS val 213614656
9 ecr 2619955275], length 0
09:53:34.192778 IP (tos 0x0, ttl 64, id 27836, offset 0, flags [DF], proto UDP (17), le
ngth 69)
    457a0067943c.37564 > metadata.google.internal.domain: 2458+ PTR? 2.0.18.172.in-addr
.arpa. (41)
09:53:34.198996 IP (tos 0x0, ttl 63, id 0, offset 0, flags [none], proto UDP (17), leng
th 140)
    metadata.google.internal.domain > 457a0067943c.37564: 2458 1/0/0 2.0.18.172.in-addr
.arpa. PTR nginx-us-east1-b.c.qwiklabs-terminal-vms-prod-00.internal. (112)
09:53:34.200507 IP (tos 0x0, ttl 64, id 62242, offset 0, flags [DF], proto UDP (17), le
ngth 74)
    457a0067943c.46058 > metadata.google.internal.domain: 59103+ PTR? 254.169.254.169.i
n-addr.arpa. (46)
5 packets captured
8 packets received by filter
0 packets dropped by kernel
```

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.**

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.

Task 3. Capture network traffic with tcpdump

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**:

```
analyst@457a0067943c:~$ sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap &
[1] 12775
analyst@457a0067943c:~$ tcpdump: listening on eth0, link-type EN10MB (Ethernet), captur
e size 262144 bytes
```

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**.

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

```
analyst@457a0067943c:~$ 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>
analyst@457a0067943c:~$ 9 packets captured
10 packets received by filter
0 packets dropped by kernel
9 packets captured
10 packets received by filter
0 packets dropped by kernel
```

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:

```
analyst@457a0067943c:~$ ls -l capture.pcap
-rw-r--r-- 1 root root 1412 Dec  8 09:57 capture.pcap
```

Task 4. Filter the captured packet data

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

```
analyst@457a0067943c:~$ sudo tcpdump -nn -r capture.pcap -v
reading from file capture.pcap, link-type EN10MB (Ethernet)
09:57:09.675805 IP (tos 0x0, ttl 64, id 38989, offset 0, flags [DF], proto TCP (6), len
gth 60)
    172.17.0.2.54676 > 142.250.98.101.80: Flags [S], cksum 0x9da1 (incorrect -> 0x2cb2)
, seq 3853973179, win 32660, options [mss 1420,sackOK,TS val 1647307240 ecr 0,nop,wscal
e 6], length 0
09:57:09.676596 IP (tos 0x0, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length
 60)
    142.250.98.101.80 > 172.17.0.2.54676: Flags [S.], cksum 0x83d6 (correct), seq 26179
2362, ack 3853973180, win 65535, options [mss 1420,sackOK,TS val 2940389141 ecr 1647307
240,nop,wscale 8], length 0
09:57:09.676616 IP (tos 0x0, ttl 64, id 38990, offset 0, flags [DF], proto TCP (6), len
gth 52)
    172.17.0.2.54676 > 142.250.98.101.80: Flags [.], cksum 0x9d99 (incorrect -> 0xb07c)
, ack 1, win 511, options [nop,nop,TS val 1647307240 ecr 2940389141], length 0
09:57:09.676706 IP (tos 0x0, ttl 64, id 38991, offset 0, flags [DF], proto TCP (6), len
gth 137)
    172.17.0.2.54676 > 142.250.98.101.80: Flags [P.], cksum 0x9dee (incorrect -> 0x1f2f
), seq 1:86, ack 1, win 511, options [nop,nop,TS val 1647307241 ecr 2940389141], length
 85: HTTP, length: 85
        GET / HTTP/1.1
        Host: opensource.google.com
        User-Agent: curl/7.64.0
        Accept: */*

09:57:09.676876 IP (tos 0x0, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length
 52)
    142.250.98.101.80 > 172.17.0.2.54676: Flags [.], cksum 0xae09 (correct), ack 86, wi
n 1051, options [nop,nop,TS val 2940389142 ecr 1647307241], length 0
09:57:09.678394 IP (tos 0x0, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length
 601)
    142.250.98.101.80 > 172.17.0.2.54676: Flags [P.], cksum 0xcb60 (correct), seq 1:550
, ack 86, win 1051, options [nop,nop,TS val 2940389143 ecr 1647307241], length 549: HTT
P, length: 549
        HTTP/1.1 301 Moved Permanently
        Location: https://opensource.google/
        X-Content-Type-Options: nosniff
        Server: sffe
        Content-Length: 223
        X-XSS-Protection: 0
        Date: Sun, 08 Dec 2024 09:27:13 GMT
        Expires: Sun, 08 Dec 2024 09:57:13 GMT
        Cache-Control: public, max-age=1800
        Content-Type: text/html; charset=UTF-8
        Age: 1796

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

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.**

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

```
analyst@457a0067943c:~$ sudo tcpdump -nn -r capture.pcap -X
reading from file capture.pcap, link-type EN10MB (Ethernet)
09:57:09.675805 IP 172.17.0.2.54676 > 142.250.98.101.80: Flags [S], seq 3853973179, win
 32660, options [mss 1420,sackOK,TS val 1647307240 ecr 0,nop,wscale 6], length 0
        0x0000:  4500 003c 984d 4000 4006 04fc ac11 0002  E..<.M@.@.......
        0x0010:  8efa 6265 d594 0050 e5b6 f6bb 0000 0000  ..be...P........
        0x0020:  a002 7f94 9da1 0000 0204 058c 0402 080a  ................
        0x0030:  622f e9e8 0000 0000 0103 0306            b/..........
09:57:09.676596 IP 142.250.98.101.80 > 172.17.0.2.54676: Flags [S.], seq 261792362, ack
 3853973180, win 65535, options [mss 1420,sackOK,TS val 2940389141 ecr 1647307240,nop,w
scale 8], length 0
        0x0000:  4500 003c 0000 4000 7e06 5f49 8efa 6265  E..<..@.~._I..be
        0x0010:  ac11 0002 0050 d594 0f9a a26a e5b6 f6bc  .....P.....j....
        0x0020:  a012 ffff 83d6 0000 0204 058c 0402 080a  ................
        0x0030:  af42 c715 622f e9e8 0103 0308            .B..b/......
09:57:09.676616 IP 172.17.0.2.54676 > 142.250.98.101.80: Flags [.], ack 1, win 511, opt
ions [nop,nop,TS val 1647307240 ecr 2940389141], length 0
        0x0000:  4500 0034 984e 4000 4006 0503 ac11 0002  E..4.N@.@.......
        0x0010:  8efa 6265 d594 0050 e5b6 f6bc 0f9a a26b  ..be...P.......k
        0x0020:  8010 01ff 9d99 0000 0101 080a 622f e9e8  ............b/..
        0x0030:  af42 c715                                .B..
09:57:09.676706 IP 172.17.0.2.54676 > 142.250.98.101.80: Flags [P.], seq 1:86, ack 1, w
in 511, options [nop,nop,TS val 1647307241 ecr 2940389141], length 85: HTTP: GET / HTTP
/1.1
        0x0000:  4500 0089 984f 4000 4006 04ad ac11 0002  E....O@.@.......
        0x0010:  8efa 6265 d594 0050 e5b6 f6bc 0f9a a26b  ..be...P.......k
        0x0020:  8018 01ff 9dee 0000 0101 080a 622f e9e9  ............b/..
        0x0030:  af42 c715 4745 5420 2f20 4854 5450 2f31  .B..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                   :.*/*....
09:57:09.676876 IP 142.250.98.101.80 > 172.17.0.2.54676: Flags [.], ack 86, win 1051, o
ptions [nop,nop,TS val 2940389142 ecr 1647307241], length 0
        0x0000:  4500 0034 0000 4000 7e06 5f51 8efa 6265  E..4..@.~._Q..be
        0x0010:  ac11 0002 0050 d594 0f9a a26b e5b6 f711  .....P.....k....
        0x0020:  8010 041b ae09 0000 0101 080a af42 c716  ............B..
        0x0030:  622f e9e9                                b/..
09:57:09.678394 IP 142.250.98.101.80 > 172.17.0.2.54676: Flags [P.], seq 1:550, ack 86,
 win 1051, options [nop,nop,TS val 2940389143 ecr 1647307241], length 549: HTTP: HTTP/1
.1 301 Moved Permanently
        0x0000:  4500 0259 0000 4000 7e06 5d2c 8efa 6265  E..Y..@.~.],..be
        0x0010:  ac11 0002 0050 d594 0f9a a26b e5b6 f711  .....P.....k....
        0x0020:  8018 041b cb60 0000 0101 080a af42 c717  .....`.......B..
        0x0030:  622f e9e9 4854 5450 2f31 2e31 2033 3031  b/..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
```

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.**