Exemplar: Capture your first packet

Lab By Google Cloud

This lab may incorporate AI tools to support your learning

Activity overview

As a security analyst, it's important to know how to capture and filter network traffic in a Linux-based environment. You'll also need to know how tools connected and interact with network interfaces.

In this lab activity, you'll perform tasks associated with using tcpdump to capture network traffic. You'll inspect the data in packet capture files and use filters to view and 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.

You'll begin your lab tasks by creating a sample packet capture file that you will use in the place of live traffic. This will give you a chance to analyze the network traffic that it contains.

Here's how you'll do this: First, you'll identify network interfaces to capture network packet data. Next, you'll use `tcpdump` to filter the network traffic. 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 this lab.*

Start your lab

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

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 need 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
```

This command returns output similar to the following:

```
ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:ac11:3ff:fe02:3  prefixlen 64  scopeid 0x20<link>
        ether 02:42:ac:11:03:02  txqueuelen 0  (Ethernet)
        RX packets 1396  bytes 12311311 (11.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1095  bytes 95888 (93.6 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
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        ether 00:00:00:00:00:00  txqueuelen 1000  (Local Loopback)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The Ethernet network interface is identified by the entry with the `ens` prefix.

By, in this lab, you'll use `ens` as the interface that you will capture network packet data from in the following steps.

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

```
sudo tcpdump -D
```

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.

Identify network interfaces

Check my progress

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.

1. Filter live network packet data from the `ens` interface with `tcpdump`:

```
sudo tcpdump -i ens4 -v -c5
```

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

- `-i ens4`: Captures data specifically from the `ens4` interface.
- `-v`: Displays detailed packet data.
- `-c5`: Captures 5 packets of data.

Now, let's see 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 ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
17:09:42.000000 IP (tos 0x0, ttl 64, id 24, offset 0, flags [DF], proto TCP (6), length 52)
    localhost.ssh > gcpdiag-ce-qwiklab-vm.ephemeral.ssh: Flags [P.], cksum 0x4a7f (incorrect -> 0x1e4f), seq 1:12, ack 1, win 501, options [nop,nop,TS val 1925 ecr 1925], length 11
17:09:42.000000 IP (tos 0x0, ttl 64, id 2455, offset 0, flags [DF], proto TCP (6), length 40)
    gcpdiag-ce-qwiklab-vm.ephemeral.ssh > localhost.ssh: Flags [.], cksum 0x4a73 (incorrect -> 0x9ed4), ack 12, win 501, options [nop,nop,TS val 1925 ecr 1925], length 0
5 packets received by filter
0 packets dropped by kernel
```

The specific packet data in your lab may look a little different under and may even be for entirely different types of network traffic. The specific details, such as system names, addresses, and port numbers, will be different. However, the overall structure and output is good for 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 end of the packet capture, `tcpdump` reported that 5 packets were received by the network interface, and it provided a summary of the first type and the capture size in bytes:

```
tcpdump: listening on ens4, 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:

```
23:24:16.934375 IP
```

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

```
(tos 0x0, ttl 64, id 5840, offset 0, flags [DF], proto TCP (6), length 104)
```

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 protocol packets.

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

```
lab2010-1700.5000 > hpins.ssh.ntt > gcplab.ntt.net: seq 1:53, ack 1, win 501, options [nop,nop,TS val 1925 ecr 1925], length 52
```

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 as `localhost`. The port number for the SSH protocol appears the second packet. In your live data, this may be different and IP address rather than a name.

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 fillers the header data for the inner TCP packet:

```
Flags [P.], cksum 0x5f83 (incorrect -> 0x3ed0), seq 1001:1053, ack 1001, win 501, options [nop,nop,TS val 1925 ecr 1925], length 52
```

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 TCP packet data fields.

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

Inspect network traffic with tcpdump

Check my progress

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 ens4 -vn -c5 port 80 -w capture.pcap &
```

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

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

- `-i ens4`: Captures data from the `ens4` interface.
- `-vn`: 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.
- `-c5`: Captures 5 packets of data.
- `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 from `tcpdump` will appear in your terminal. The text will not affect the command you've typed, but feel free to stop for the rest of the lab.

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

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

When the `curl` command is used the 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
```

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.

Capture network traffic with tcpdump

Check my progress

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
```

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

- `-nn`: Disable port and protocol name lookup.
- `-r capture.pcap`: Read from the capture 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)
17:09:42.000000 IP (tos 0x0, ttl 64, id 24, offset 0, flags [DF], proto TCP (6), length 52)
    172.17.0.2.5000 > 169.76.63.80.80: Flags [S], cksum 0x4a7f (incorrect -> 0x1e4f), seq 1:12, ack 1, win 501, options [nop,nop,TS val 1925 ecr 1925], length 11
17:09:42.000000 IP (tos 0x0, ttl 64, id 2455, offset 0, flags [DF], proto TCP (6), length 40)
    169.76.63.80.80 > 172.17.0.2.5000: Flags [.], cksum 0x4a73 (incorrect -> 0x9ed4), ack 12, win 501, options [nop,nop,TS val 1925 ecr 1925], length 0
5 packets received by filter
0 packets dropped by kernel
```

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
```

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 use this option to view packet data 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 to 9 and the 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 or digital data.*

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

Filter the captured packet data

Check my progress

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 5 packets on any interface with the verbose option?

- sudo tcpdump -ND -c5 -v
- sudo tcpdump -vD -c5 -any -v
- sudo tcpdump -D -c5 -v
- sudo tcpdump -c5 -any -v

Submit

Answer: Use the `sudo tcpdump -D -c5 -any -v`.

What does the `-c` option indicate?

- Incremental monitoring mode
- Capture incoming packets only
- The number of packets to capture
- The network interface to monitor

Submit

Answer: The `-c` option indicates the network interface to monitor.

What type of information does the `-v` option include?

- Virtual packets
- Verbose information
- Version information
- Packets including the letter 'V'

Submit

Answer: The `-v` option provides verbose information.

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

- sudo tcpdump
- sudo -v
- sudo tcpdump -D
- sudo capture p cap

Submit

Answer: Use the `sudo tcpdump -D` command.

Conclusion

Great work!

You have gained practical experience to enable you to:

- identify network interfaces
- use `tcpdump` to capture live packet network data for inspection,
- interpret the information that `tcpdump` outputs regarding a packet, and
- save and read packet data for later analysis.

You're well on your way to capturing your first packet.

End your lab

Before you end this lab, make sure you've satisfied that you've completed all the tasks, and follow these steps:

1. Click **End Lab**. A pop-up box will appear. Click **Submit** to confirm that you've done. Ending the lab will remove your access to the Qwiklabs shell. You won't be able to access the work you've completed in it again.
2. Another pop-up box will ask you to rate the lab and provide feedback comments. You can complete that if you choose to.
3. Close the browser tab containing the lab to return to your course.
4. Refresh the browser tab for the course to mark the lab as complete.