Activity: Capture your first packet

Introduction

In this lab activity, you'll capture and analyze live network traffic using tcpdump. You'll use Linux commands in the Bash shell to complete these steps.

What you'll do

You have multiple tasks in this lab:

- Identify available network interfaces
- Use tcpdump to capture live network traffic
- Save network traffic to a packet capture file
- Filter the packet capture data

Activity: Capture your first packet

1 hourFree

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.

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!

content_copy

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!

content_copy

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.

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

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

Copied!

content_copy

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.

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

Copied!

content_copy

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!

content_copy

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!

content_copy

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
```

Copied!

content_copy

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!

content_copy

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.

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 3 packets on any interface with the verbose option?

sudo tcpdump -N2 -i any -v

sudo tcpdump -s3 -i all -v

sudo tcpdump -n3 -i any -v

sudo tcpdump -c3 -i any -v

Submit
What does the -i option indicate?

The network interface to monitor

The number of packets to capture

Capture incoming packets only

Incremental monitoring mode

Submit

What type of information does the -v option include?

Packets including the letter `V`

Verbose information

Virtual packets

Version information

Submit

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

sudo tcpdump

sudo tcpdump -D

sudo ls

sudo capture p.cap

Submit

Conclusion

Great work!

You have gained practical experience to enable you to

- identify network interfaces,
- use the `tcpdump` command to capture network data for inspection,
- interpret the information that `tcpdump` outputs regarding a packet, and

- save and load packet data for later analysis.

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

End your lab

Before you end the lab, make sure you’re 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're done. Ending the lab will remove your access to the Bash 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 this 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.

Exemplar: Capture your first packet

1 hourFree

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.

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!

content_copy

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!

content_copy

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.

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

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

Copied!

content_copy

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.

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

Copied!

content_copy

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!

content_copy

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!

content_copy

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
```

Copied!

content_copy

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!

content_copy

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.

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 3 packets on any interface with the verbose option?

sudo tcpdump -c3 -i any -v

sudo tcpdump -s3 -i all -v

sudo tcpdump -N2 -i any -v

sudo tcpdump -n3 -i any -v

Submit

Answer: Use the `sudo tcpdump -c3 -i any -v`.

What does the -i option indicate?

Capture incoming packets only

The network interface to monitor

The number of packets to capture

Incremental monitoring mode

Submit

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

What type of information does the -v option include?

Version information

Verbose information

Packets including the letter `V`

Virtual packets

Submit

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

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

sudo tcpdump

sudo capture p.cap

sudo tcpdump -D

sudo ls

Submit

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

Conclusion

Great work!

You have gained practical experience to enable you to

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

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

End your lab

Before you end the lab, make sure you’re 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're done. Ending the lab will remove your access to the Bash 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 this 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.