# Exemplar: Capture your first packet

**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!

This exemplar is a walkthrough of the previous Qwiklab activity, including detailed instructions and solutions. You may use this exemplar if you were unable to complete the lab and/or you need extra guidance in competing lab tasks. You may also refer to this exemplar to prepare for the graded quiz in this module.

**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**

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:

A white rectangular object with blue lines

AI-generated content may be incorrect.

This command returns output similar to the following:

A screenshot of a computer

AI-generated content may be incorrect.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:

A white rectangular object with blue lines

AI-generated content may be incorrect.

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**

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**:

A white rectangular object with blue lines

AI-generated content may be incorrect.

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:

A screen shot of a computer

AI-generated content may be incorrect.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:

A screen shot of a computer

AI-generated content may be incorrect.

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

A white rectangular object with blue lines

AI-generated content may be incorrect.

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:

A screen shot of a computer

AI-generated content may be incorrect.

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:

A screen shot of a computer

AI-generated content may be incorrect.

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:

A screenshot of a computer

AI-generated content may be incorrect.

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 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**:

A screen shot of a computer

AI-generated content may be incorrect.

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:

A white rectangular object with a blue border

AI-generated content may be incorrect.

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:

A white rectangular object with blue lines

AI-generated content may be incorrect.

***Note:****The "Done" in the output indicates that the packet was captured.*

**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:

A white rectangular box with text

AI-generated content may be incorrect.

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:

A screen shot of a computer code

AI-generated content may be incorrect.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:

A white rectangular object with red and blue text

AI-generated content may be incorrect.

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.*

**Test your understanding**

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

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

What does the -i option indicate?

The **-i**option indicates the network interface to monitor.

What type of information does the -v option include?

**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?

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.