

Oct 13, 2022 • 5 min read

Using tcpdump Command in Linux to Analyze Network



Team LHB

Table of Contents



1. Checking the Available Interfaces
2. Capturing Packets for a Specific interface
3. Presetting Capture Count
4. Getting a Verbose Output
5. Printing the Captured Data in ASCII Format
6. Capturing Packets Sent From a Specific Source IP
7. Capturing Packets Sent to a Specific Destination IP
8. Using Filtering Options with Tcpdump
 - Port number
 - Protocol
 - Host Filter
9. Saving the Captured Data
10. Reading the Captured Data
- Conclusion

Tcpdump is a great tool for analyzing networks and hunting down associated network problems. It captures packets as they go by and shows you what's going on and coming in on your network. The output from the command displays on the STDOUT and can also be stored in a file.

Thanks to the developers, who have kept the Tcpdump as an open source project. It is freely available on Unix and Linux systems. Windows has a 'Microolap TCPDUMP for Windows' variant with an associated price tag.

`tcpdump` has a long list of options available for use. In this article, I'll focus on core options that are frequently used.

Subscribe

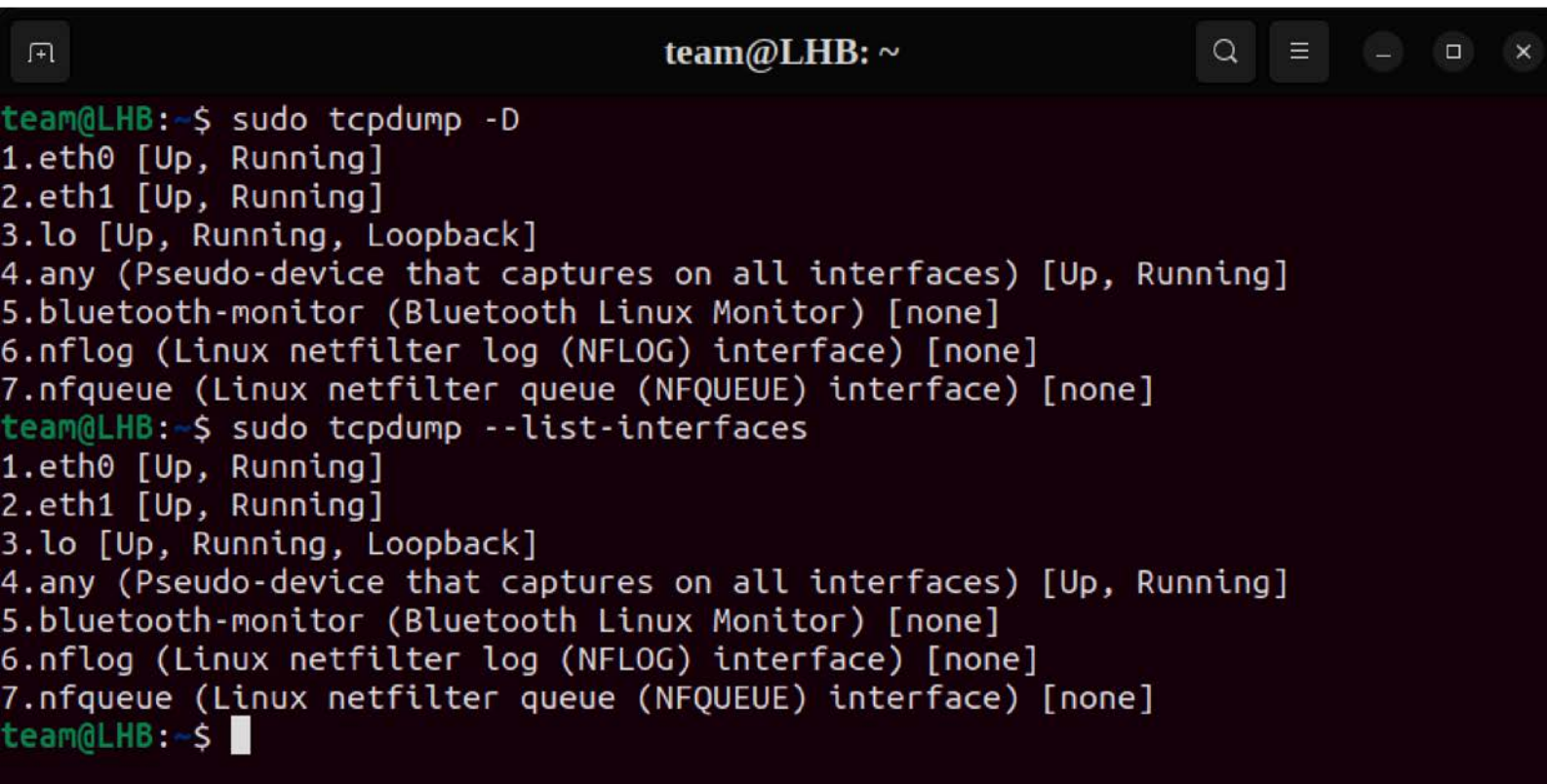
1. Checking the Available Interfaces

To check all the available interfaces to capture on, use the '-D' flag as:

```
sudo tcpdump -D
```

This will list all the interfaces on the system including wireless and wired interfaces and others. The same functionality can also be gained with the `--list-interfaces` flag:

```
sudo tcpdump --list-interfaces
```



```
team@LHB: ~  
team@LHB:~$ sudo tcpdump -D  
1.eth0 [Up, Running]  
2.eth1 [Up, Running]  
3.lo [Up, Running, Loopback]  
4.any (Pseudo-device that captures on all interfaces) [Up, Running]  
5.bluetooth-monitor (Bluetooth Linux Monitor) [none]  
6.nflog (Linux netfilter log (NFLOG) interface) [none]  
7.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]  
team@LHB:~$ sudo tcpdump --list-interfaces  
1.eth0 [Up, Running]  
2.eth1 [Up, Running]  
3.lo [Up, Running, Loopback]  
4.any (Pseudo-device that captures on all interfaces) [Up, Running]  
5.bluetooth-monitor (Bluetooth Linux Monitor) [none]  
6.nflog (Linux netfilter log (NFLOG) interface) [none]  
7.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]  
team@LHB:~$
```

2. Capturing Packets for a Specific interface

Without using any option, Tcpcdump will scan all the interfaces. The `-i` flag captures traffic from a specific interface:

```
tcpdump -i <target-interface>
```

Replace the `target-interface` with the name of the interface you want to scan. For example, in the case of the interface `eth0`, this command will be as:

```
sudo tcpdump -i eth0
```

Subscribe

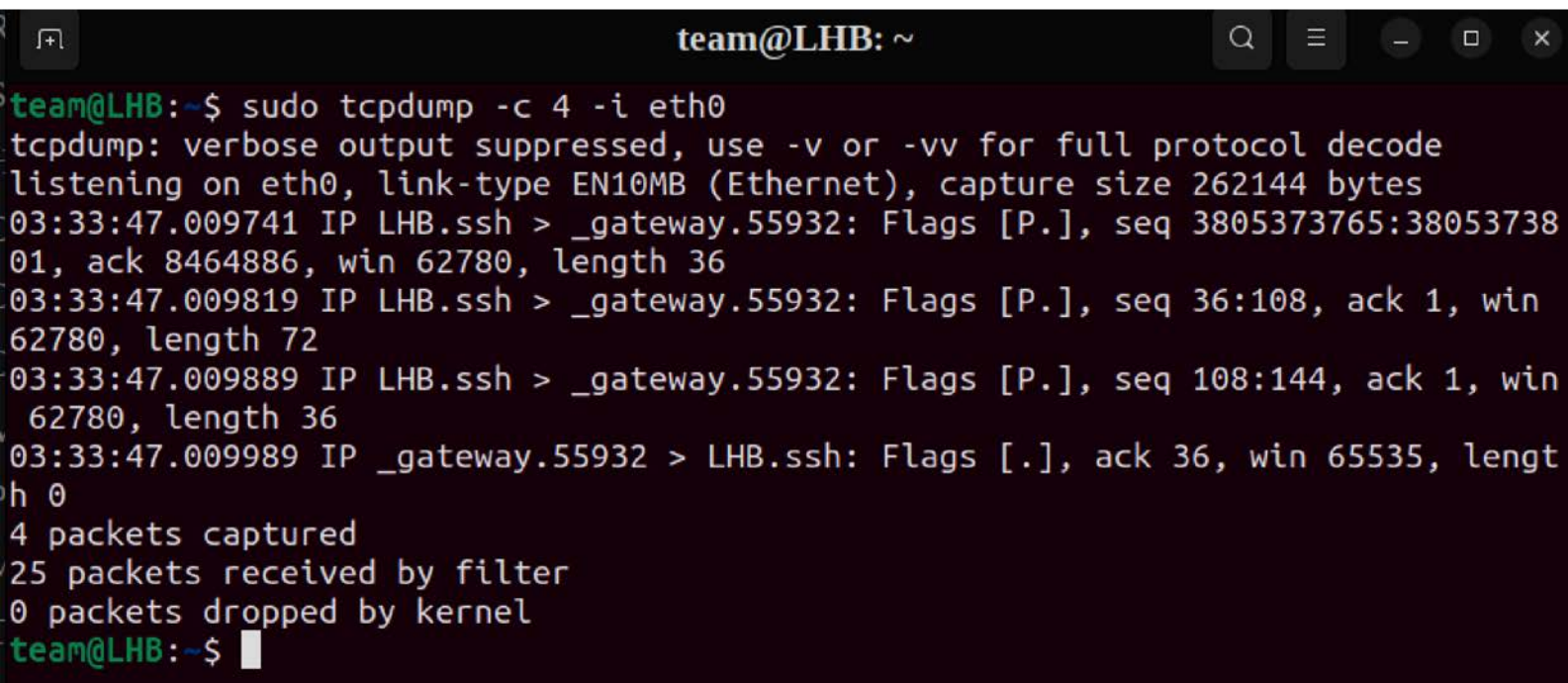
Note: From now on, I'll use the `eth0` or `eth1` as the target interface. So wherever you see the `-i` flag, it will be accompanied by either the interface `eth0` or `eth1`.

3. Presetting Capture Count

The `-c` flag can be used to preset the number of packets to be captured.

As an example, let's set this value to 4 for capturing four packets. The command, in this case, will be:

```
sudo tcpdump -c 4 -i eth0
```

A terminal window titled 'team@LHB: ~' with search, menu, and window control icons. The command 'sudo tcpdump -c 4 -i eth0' is executed. The output shows 'tcpdump: verbose output suppressed, use -v or -vv for full protocol decode', 'listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes', and four captured packets from LHB.ssh to _gateway.55932. It concludes with '4 packets captured', '25 packets received by filter', and '0 packets dropped by kernel'.

```
team@LHB:~$ sudo tcpdump -c 4 -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
03:33:47.009741 IP LHB.ssh > _gateway.55932: Flags [P.], seq 3805373765:3805373801, ack 8464886, win 62780, length 36
03:33:47.009819 IP LHB.ssh > _gateway.55932: Flags [P.], seq 36:108, ack 1, win 62780, length 72
03:33:47.009889 IP LHB.ssh > _gateway.55932: Flags [P.], seq 108:144, ack 1, win 62780, length 36
03:33:47.009989 IP _gateway.55932 > LHB.ssh: Flags [.], ack 36, win 65535, length 0
4 packets captured
25 packets received by filter
0 packets dropped by kernel
team@LHB:~$
```

If you do not specify a count, the capture operation is to be manually interrupted using the key combination `ctrl+c` or `ctrl+z`.

In the following article, I'll add the `-c` flag with other flags wherever required. This will help us to clearly and easily understand the output of a command.

4. Getting a Verbose Output

For getting a verbose output of a `tcpdump` command, you can use the `-v` flag:

```
sudo tcpdump -c 6 -v -i eth0
```

Subscribe

You can further increase the level of verbosity using more `-v` flags as `-vv` or `-vvv`.

`vvv`). This will yield more detailed output on the terminal:

```
sudo tcpdump -vv -i eth0
```

5. Printing the Captured Data in ASCII Format

Sometimes we may require the Tcpdump output to be in HEX or ASCII format. We can work out this using the options `-A` for ASCII format and `-XX` for both ASCII and HEX format:

```
sudo tcpdump -XX -i eth0
```

```
team@LHB:~$ sudo tcpdump -XX -i eth0 -c 5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
03:44:47.313927 IP LHB.ssh > _gateway.55932: Flags [P.], seq 3813265565:3813265601, ack 8474838, win 62780, length 36
    0x0000: 5254 0012 3502 0800 27a2 6bfd 0800 4510  RT..5...'.k...E.
    0x0010: 004c fe88 4000 4006 2403 0a00 020f 0a00  .L..@.@.$.....
    0x0020: 0202 0016 da7c e349 d09d 0081 50d6 5018  ....|.I....P.P.
    0x0030: f53c 184f 0000 65ff cbf1 895c a043 c6cb  .<.O..e....\C..
    0x0040: 7351 696a 9264 7812 2140 af01 e1b3 6833  sQij.dx.!@....h3
    0x0050: dfcb e3ec 21d8 1abb 6f8b                ....!...o.
03:44:47.314156 IP _gateway.55932 > LHB.ssh: Flags [.], ack 36, win 65535, length 0
    0x0000: 0800 27a2 6bfd 5254 0012 3502 0800 4500  ..'.k.RT..5...E.
```

6. Capturing Packets Sent From a Specific Source IP

In case you want to inspect the traffic coming from a specific source IP address, use this command:

```
sudo tcpdump -i eth0 src <source-ip-address>
```

Let's take the source IP as `192.168.56.11` and see the details of the traffic:

```
sudo tcpdump -i eth1 -c 5 src 192.168.56.11
```

Subscribe


```
team@LHB:~$ sudo tcpdump -i eth1 -c 5 src 192.168.56.11
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
04:00:33.121335 IP 192.168.56.11.47094 > LHB.ssh: Flags [S], seq 3323970319, win
 64240, options [mss 1460,sackOK,TS val 4169240945 ecr 0,nop,wscale 7], length 0
04:00:33.121700 IP 192.168.56.11.47094 > LHB.ssh: Flags [.), ack 82719130, win 5
02, options [nop,nop,TS val 4169240945 ecr 2585815412], length 0
04:00:33.121966 IP 192.168.56.11.47094 > LHB.ssh: Flags [P.), seq 0:41, ack 1, w
in 502, options [nop,nop,TS val 4169240946 ecr 2585815412], length 41
04:00:33.130722 IP 192.168.56.11.47094 > LHB.ssh: Flags [.), ack 42, win 502, op
```

The count 5 here will capture only the first five packets.

7. Capturing Packets Sent to a Specific Destination IP

In case you want to inspect the traffic sent to a specific destination IP address, use the command:

```
sudo tcpdump -i eth0 dst <source-ip-address>
```

Let's take the destination IP as `192.168.56.11` and see the details of the traffic:

```
sudo tcpdump -i eth1 -c 5 dst 192.168.56.11
```

```
team@LHB:~$ sudo tcpdump -i eth1 dst 192.168.56.11
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
04:02:31.474545 IP LHB.ssh > 192.168.56.11.47096: Flags [F.), seq 1801154518, ac
k 3356396022, win 501, options [nop,nop,TS val 2585933765 ecr 4169359297], lengt
h 0
04:02:42.866436 IP LHB.ssh > 192.168.56.11.47098: Flags [S.), seq 1932475958, ac
k 122598727, win 65160, options [mss 1460,sackOK,TS val 2585945157 ecr 416937069
0,nop,wscale 7], length 0
04:02:42.867200 IP LHB.ssh > 192.168.56.11.47098: Flags [.), ack 42, win 509, op
tions [nop,nop,TS val 2585945158 ecr 4169370691], length 0
04:02:42.874872 IP LHB.ssh > 192.168.56.11.47098: Flags [P.), seq 1:42, ack 42,
```

8. Using Filtering Options with Tcpdump

It's a good approach to narrow down your captured data for inspection. This will eliminate unnecessary traffic and simplify your job. You can do this by filtering traffic based on host, ports, protocols, and other criteria.

Let's see some of them:

Subscribe

Port number

In case you want to filter traffic based on port number, say port 22, then execute the `tcpdump` command as:

```
sudo tcpdump -i eth0 port 22
```

This command will capture both the TCP and UDP traffic.

Protocol

Similar to the port directive, the `proto` directive filters the packet capture based on particular traffic. Here, you can either use the protocol name or the protocol number as the argument value:

```
sudo tcpdump -i eth0 proto tcp
```

```
sudo tcpdump -i eth0 proto 6
```

To your surprise, the two commands above are equivalent. It is because `6` is the protocol number for TCP.

Host Filter

The host argument simply filters the traffic from a specific host using its IP:

```
sudo tcpdump -i eth0 host 192.168.56.10
```

This will capture all the traffic and out from this host. Interestingly, you can apply multiple filters to your host to target a specific type of packet traffic.

For example:

```
sudo tcpdump -i eth1 -c 50 "(host 192.168.56.11) and (port 443 or port 80)"
```

Subscribe

```

team@LHB:~$ sudo tcpdump -i eth1 -c 50 "(host 192.168.56.11) and (port 443 or port 80)"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
04:13:18.798950 IP 192.168.56.11.35174 > LHB.http: Flags [S], seq 4279187927, win 64240, options [mss 1460,sackOK,TS val 4170006622 ecr 0,nop,wscale 7], length 0
04:13:18.798970 IP LHB.http > 192.168.56.11.35174: Flags [R.], seq 0, ack 4279187928, win 0, length 0
04:13:43.576525 IP 192.168.56.11.35176 > LHB.http: Flags [S], seq 4218852565, win 64240, options [mss 1460,sackOK,TS val 4170031400 ecr 0,nop,wscale 7], length 0
04:13:43.576551 IP LHB.http > 192.168.56.11.35176: Flags [R.], seq 0, ack 4218852566, win 0, length 0
04:14:06.524071 IP 192.168.56.11.50024 > LHB.https: Flags [S], seq 2634007850, win 64240, options [mss 1460,sackOK,TS val 4170054347 ecr 0,nop,wscale 7], length 0
04:14:06.524092 IP LHB.https > 192.168.56.11.50024: Flags [R.], seq 0, ack 2634007851, win 0, length 0

```

Here, I have merged different filter rules into a single rule. You can see this rule is filtering `http` and `https` traffic. This is because the rule contains the filter for ports 80 and 443, the [common networking ports](#).

9. Saving the Captured Data

If you want to store the captured data in a file, you can do it like this.

```
sudo tcpdump -i eth0 -c 10 -w my_capture.pcap
```

```

team@LHB:~$ sudo tcpdump -i eth0 -c 10 -w my_capture.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C2 packets captured
3 packets received by filter
0 packets dropped by kernel

```

Keep the packet count to a smaller value; otherwise, you may have to stop the process manually.

10. Reading the Captured Data

You can use the data stored in the `.pcap` file for analysis with [Wireshark](#) or any other graphical network protocol analyzer.

You can read it with `tshark` itself

Subscribe

You can read it with tcpdump tool.

```
tcpdump -r my_capture.pcap
```

```
team@LHB:~$ tcpdump -r my_capture.pcap
reading from file my_capture.pcap, link-type EN10MB (Ethernet)
04:32:32.901951 IP LHB.ssh > _gateway.55932: Flags [P.], seq 3814439673:3814439709, ack 8504478, win 62780, length 36
04:32:32.902234 IP _gateway.55932 > LHB.ssh: Flags [.], ack 36, win 65535, length 0
team@LHB:~$
```

The above screenshot shows the data of the above `my_capture.pcap` file.

Conclusion

That's all for now. I hope you have a good idea of how to use different ways to work with tcpdump command. It is the best option when you are capturing packets from a remote headless machine.

If you want a more visual way to understand packet capture, try Wireshark.

Commands

SHARE



Subscribe