

How to Install tcpdump

The most straightforward way to install *tcpdump* is via your Linux system's package manager. Fortunately, *tcpdump* is available by default on most Linux distributions.

Debian / Ubuntu

```
sudo apt install tcpdump
```

AlmaLinux / CentOS Stream / Fedora / Rocky Linux

```
sudo dnf install tcpdump
```

You can verify your installation using the command below:

```
sudo tcpdump --version
```

The command's output may vary, but it should be similar to what you see here:

```
tcpdump version 4.9.3
libpcap version 1.9.1 (with TPACKET_V3)
OpenSSL 1.1.1k FIPS 25 Mar 2021
```

How to Interpret tcpdump Output

Before getting started using *tcpdump*, it can be useful to familiarize yourself with how *tcpdump* displays packets. In *tcpdump*, each line corresponds to a single packet, displayed in distinct sections. Here is an example from later on in this tutorial:

```
20:26:16.902555 IP example.hostname-one.com.ssh > example.hostname-two.com.6402
```



Each packet's display breaks down like this, using the example above as a guide:

- `20:26:16.902555` is the Unix timestamp, representing when the packet was sent. Refer to the **How to Use Advanced Display Options with tcpdump** section further on to see how you can display more human-readable timestamps instead.
- `IP` is the network protocol. In this case, the packet is a TCP packet, which show up under the `IP` designation. An ICMP packet, by contrast, would show `ICMP` here.
- `example.hostname-one.com.ssh > example.hostname-two.com.64024` are the source and destination hostnames. The source is before the `>` symbol, and the destination is after.

The last entry on each host above, `.ssh` for the source and `.64024` for the destination, is the port.

Refer to the **How to Use Advanced Display Options with tcpdump** section below to see how to display host IP addresses instead of hostnames.

- The rest is query information.

In the above, `Flag` indicates the TCP flag. The flags for TCP packets can be any combination of the following. Most often, the flag field contains a combination of one of these letter designations and the `.` symbol (like the `P.` in the example above):

- `S` for SYN
- `F` for FIN
- `P` for PUSH
- `R` for RST
- `U` for URG
- `w` for ECN CWR
- `E` for ECN-Echo
- `.` for ACK
- No flag, indicating that no flag is set for the packet

Explanations of each flag can be found in the Wikipedia [page on TCP](#).

DNS queries, by contrast, replace the `Flag` portion with the DNS query ID, something like `44000+`.

How to Start Reading Network Traffic with tcpdump

These next sections show you how to get started using *tcpdump* for capturing and analyzing network traffic.

The example commands and output in these sections use `example.hostname-one.com` (also as `192.0.2.0`) and `example.hostname-two.com` (also as `198.51.100.0`) for hosts throughout

this tutorial. Replace these with your own relevant hostnames/IP addresses when reproducing these examples.

Finding Interfaces

tcpdump can provide a list of available network interfaces on your Linux system. In the next sections, these can be used to determine where you want to listen for packets.

To get a list of available interfaces, use the *tcpdump* command with the `-D` option:

```
sudo tcpdump -D
```

```
1.eth0 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up, Running]
4.bluetooth-monitor (Bluetooth Linux Monitor) [none]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.usbmon0 (Raw USB traffic, all USB buses) [none]
```

Capturing Packets

You can begin capturing packets by simply executing the *tcpdump* command. By default, the command uses the lowest-numbered interface, which you can see from the command above would be `eth0`. However, it can be good practice to explicitly provide the interface, which you can accomplish with the `-i` option.

```
sudo tcpdump -i eth0
```

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:45:52.001714 IP example.hostname-one.com.ssh > example.hostname-two.com.4916
```

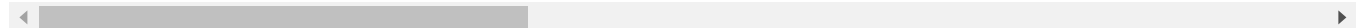
```
16:45:52.002503 IP example.hostname-one.com.47687 > resolver08.atlanta.linode.c
16:45:52.033269 IP example.hostname-two.com.49165 > example.hostname-one.com.s
16:45:52.118992 IP resolver08.atlanta.linode.com.domain > example.hostname-one.
16:45:52.119259 IP example.hostname-one.com.48245 > resolver08.atlanta.linode.c
[...]
```

You can stop capturing packets using the *Ctrl + C* key combination. And, as you may notice, the output from *tcpdump* can accumulate quickly. This is especially the case for broad, unfiltered searches like the one above.

You can make the output more manageable for the purposes of getting to know *tcpdump* by using the *-c* option. This option allows you to set a number of lines (packets) of output:

```
sudo tcpdump -i eth0 -c 5
```

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:45:50.085053 IP example.hostname-one.com.ssh > example.hostname-two.com.6402
19:45:50.086314 IP example.hostname-one.com.ssh > example.hostname-two.com.6402
19:45:50.086463 IP example.hostname-one.com.ssh > example.hostname-two.com.6402
19:45:50.086859 IP example.hostname-one.com.53728 > resolver06.atlanta.linode.c
19:45:50.087197 IP resolver06.atlanta.linode.com.domain > example.hostname-one.
5 packets captured
12 packets received by filter
0 packets dropped by kernel
```



tcpdump also gives you the option of capturing packets on all available interfaces, by specifying *any* as the interface. Doing so lets you cast an even wider net when observing network traffic:

```
sudo tcpdump -i any
```

Applying Filters

By itself, *tcpdump* produces a lot of output. It can accumulate quickly, making it difficult to find what you are looking for. That is why, most of the time, you are likely to want to

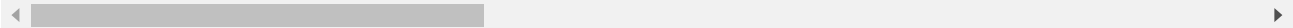
use *tcpdump* with specific filters. These allow you to limit the captured packets based on certain criteria, keeping out many of the results that you do not need.

Here, you can see the primary filtering options available in *tcpdump*. In the next section, you can also see how to combine filters and use negative filters, allowing you to even further refine your packet capturing.

- Port filtering can be accomplished using the `port` option followed by a specific port you want to observe network traffic on:

```
sudo tcpdump -i eth0 -c 1 port 80
```

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:12:53.530741 IP example.hostname-two.com.64251 > example.hostname-one.com.80
1 packet captured
1 packet received by filter
0 packets dropped by kernel
```



- Host filtering can be accomplished using the `host` option followed by the IP address for a host you want to observe network traffic for:

```
sudo tcpdump -i eth0 -c 5 host 170.187.150.148
```

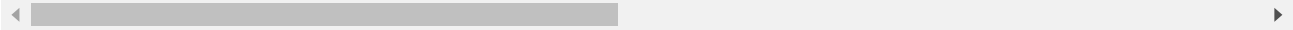
```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:20:18.850561 IP example.hostname-one.com.ssh > example.hostname-two.com.22
20:20:18.850695 IP example.hostname-one.com.ssh > example.hostname-two.com.22
20:20:18.851206 IP example.hostname-one.com.43356 > resolver06.atlanta.linode.com.53
20:20:18.851759 IP resolver06.atlanta.linode.com.domain > example.hostname-one.com.53
20:20:18.852052 IP example.hostname-one.com.54556 > resolver06.atlanta.linode.com.53
5 packets captured
10 packets received by filter
0 packets dropped by kernel
```



- Protocol filtering can be used to narrow results to only a given protocol type. You can accomplish this by simply appending the protocol designation to the *tcpdump* command:

```
sudo tcpdump -i eth0 -c 5 icmp
```

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:21:13.452381 IP example.hostname-one.com > zg-0421d-148.stretchoid.com: :
20:21:18.979723 IP example.hostname-one.com > 45.155.204.63: ICMP host exam
20:21:32.953615 IP example.hostname-one.com > 45.143.203.88: ICMP host exam
20:21:36.452595 IP ec2-54-197-15-194.compute-1.amazonaws.com > example.hosti
20:21:36.452732 IP example.hostname-one.com > ec2-54-197-15-194.compute-1.ai
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```



Combining Filters

Filters can be made even more effective by combining them using the logical operators `and`, `or`, and `not`. Here's an example that combines the host and port options:

```
sudo tcpdump -i eth0 -c 5 host 192.0.2.0 and port 80
```

The `not` operator can be used on its own as well:

```
sudo tcpdump -i eth0 -c 5 not port 80
```

Saving Results

Often, the results from *tcpdump* are extensive, and sometimes you may want to save the results for deeper analysis at a later time. For that, you can use the *tcpdump* feature for saving results to a file.

This uses the `-w` option followed by the name of the file to save the results to:

```
sudo tcpdump -i eth0 -c 5 -w example-packet-dump.pcap
```

You can then read the results again right in *tcpdump*, using the `-r` option:

```
sudo tcpdump -r example-packet-dump.pcap
```

```
reading from file example-packet-dump.pcap, link-type EN10MB (Ethernet)
dropped privs to tcpdump
20:26:16.902555 IP example.hostname-one.com.ssh > example.hostname-two.com.64024: Flags [P], seq 123456789, win 0, len 0
20:26:16.923433 IP 107.189.4.117.51676 > example.hostname-one.com.personal-ager: Flags [P], seq 123456789, win 0, len 0
20:26:16.923532 IP example.hostname-one.com > 107.189.4.117: ICMP host example.hostname-one.com unreachable: host unreachable
20:26:16.932689 IP example.hostname-two.com.64024 > example.hostname-one.com.ssh: Flags [P], seq 123456789, win 0, len 0
20:26:17.357029 IP6 example.hostname-one.com > ff02::16: HBH ICMP6, multicast 1
```

How to Use Advanced Display Options with tcpdump

tcpdump provides a number of options to control how results display. These range from making output easier to read to providing additional information about packets.

- Use `-v` for verbose results, providing additional information for each packet. You can make the results even more verbose with `-vv`:

```
sudo tcpdump -i eth0 -c 5 -vv
```

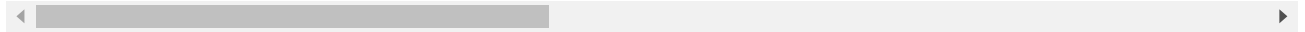
```
dropped privs to tcpdump
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144
20:27:34.957840 IP (tos 0x48, ttl 64, id 13252, offset 0, flags [DF], proto TCP, len 0) example.hostname-one.com.ssh > example.hostname-two.com.64024: Flags [P], seq 123456789, win 0, len 0
20:27:34.958582 IP (tos 0x0, ttl 64, id 59922, offset 0, flags [DF], proto ICMP, len 28) example.hostname-one.com.47790 > resolver06.atlanta.linode.com.domain: ICMP host unreachable
20:27:34.958655 IP (tos 0x48, ttl 64, id 13253, offset 0, flags [DF], proto TCP, len 0) example.hostname-one.com.ssh > example.hostname-two.com.64024: Flags [P], seq 123456789, win 0, len 0
20:27:34.960232 IP (tos 0x0, ttl 59, id 23535, offset 0, flags [none], proto ICMP, len 28) resolver06.atlanta.linode.com.domain > example.hostname-one.com.47790: ICMP host unreachable
20:27:34.960494 IP (tos 0x0, ttl 64, id 59923, offset 0, flags [DF], proto ICMP, len 28) example.hostname-one.com.47790 > resolver06.atlanta.linode.com.domain: ICMP host unreachable
```

```
example.hostname-one.com.53590 > resolver06.atlanta.linode.com.domain:
5 packets captured
9 packets received by filter
0 packets dropped by kernel
```

- Sometimes results are easier to interpret when hostnames are rendered as IP addresses. Alternatively, rendering hostnames as IP addresses may be essential when DNS resolution is unavailable. In these cases, you can use the `-n` option. This option shows hosts by their IP addresses:

```
sudo tcpdump -i eth0 -n -c 5
```

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:15:28.908978 IP6 fe80::f03c:92ff:fe4d:2fb0 > ff02::16: HBH ICMP6, multi
20:15:28.918797 IP 192.0.2.0.ssh > 198.51.100.0.64024: Flags [P.], seq 2957!
20:15:28.918915 IP 192.0.2.0.ssh > 198.51.100.0.64024: Flags [P.], seq 188:
20:15:28.919162 IP 192.0.2.0.ssh > 198.51.100.0.64024: Flags [P.], seq 352:
20:15:28.919278 IP 192.0.2.0.ssh > 198.51.100.0.64024: Flags [P.], seq 748:
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```



- To get more human-readable timestamps, as opposed to the default Unix-formatted timestamps, you can use the `-tttt` option:

```
sudo tcpdump -i eth0 -c 5 -tttt
```

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
2022-06-09 20:28:27.636762 IP example.hostname-one.com.ssh > example.hostna
2022-06-09 20:28:27.636812 IP example.hostname-one.com.ssh > example.hostna
2022-06-09 20:28:27.637105 IP example.hostname-one.com.35506 > resolver06.a
2022-06-09 20:28:27.638663 IP resolver06.atlanta.linode.com.domain > exampl
2022-06-09 20:28:27.638766 IP example.hostname-one.com.33127 > resolver06.a
5 packets captured
11 packets received by filter
0 packets dropped by kernel
```

