

Using tcpdump to analyze traffic

Tcpdump is a network utility used to capture incoming and outgoing traffic. Here's everything you need to know about using tcpdump on Linux.

Linux comes with a wide variety of network utilities to choose from. Tcpdump is a powerful network tool that can capture and analyze network traffic if you need to troubleshoot network problems on Linux.

Let's practice with the tcpdump command and explore how to use it to capture network traffic.

Install tcpdump in Linux

Tcpdump comes pre-installed with all mainstream Linux distributions and security-based alternatives, so you should be able to use it right away by typing tcpdump with the sudo prefix.

In case you are unable to run the tcpdump command and stuck at the error "*tcpdump: command not found*", let's learn how to install tcpdump on Linux machine.

To install tcpdump, fire up a terminal and run the command corresponding to the Linux distribution you are currently using:

On Debian/Ubuntu derivatives, run:

```
sudo pacman -S tcpdump
```

On Arch-based systems, run:

```
sudo pacman -S tcpdump
```

To install the tcpdump utility on Fedora, CentOS, and RHEL, issue the following command:

```
sudo dnf install tcpdump
```

Note that if you are asked to install libcap, enter **Yes** or **Y** as this is a core dependency, otherwise tcpdump will refuse to start. This will install the tcpdump utility and resolve the "*command not found*" error .

Now that tcpdump is installed on your system, let's explore the different options and functionalities it offers.

Capture network traffic with tcpdump

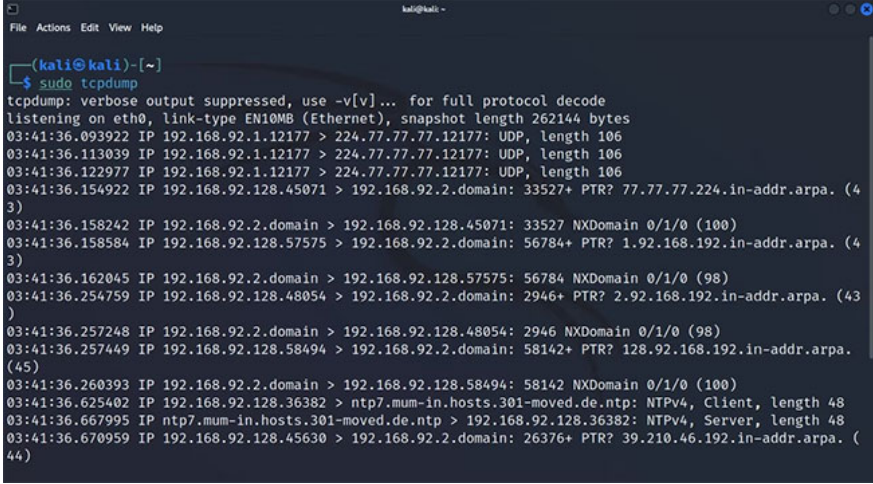
Tcpdump provides a lot of flags to modify its execution but it can also be run as a standalone command. However, running tcpdump without any flags or arguments will miss its full potential. It is better to use some flags to tweak the execution and output as needed.

Enter this command to monitor network transmissions using tcpdump:

```
sudo tcpdump
```

Now, tcpdump will start automatically capturing network packets until a interrupt signal is sent with the **Ctrl + Z** shortcut , interrupting the process manually. To limit the total number of packets captured, use the **-c** flag and enter the desired packet limit next to it:

```
sudo tcpdump -c 5
```



```
(kali㉿kali)-[~]
└─$ sudo tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
03:41:36.093922 IP 192.168.92.1.12177 > 224.77.77.77.12177: UDP, length 106
03:41:36.113039 IP 192.168.92.1.12177 > 224.77.77.77.12177: UDP, length 106
03:41:36.122977 IP 192.168.92.1.12177 > 224.77.77.77.12177: UDP, length 106
03:41:36.154922 IP 192.168.92.128.45071 > 192.168.92.2.domain: 33527+ PTR? 77.77.77.224.in-addr.arpa. (43)
03:41:36.158242 IP 192.168.92.2.domain > 192.168.92.128.45071: 33527 NXDomain 0/1/0 (100)
03:41:36.158584 IP 192.168.92.128.57575 > 192.168.92.2.domain: 56784+ PTR? 1.92.168.192.in-addr.arpa. (43)
03:41:36.162045 IP 192.168.92.2.domain > 192.168.92.128.57575: 56784 NXDomain 0/1/0 (98)
03:41:36.254759 IP 192.168.92.128.48054 > 192.168.92.2.domain: 2946+ PTR? 2.92.168.192.in-addr.arpa. (43)
03:41:36.257248 IP 192.168.92.2.domain > 192.168.92.128.48054: 2946 NXDomain 0/1/0 (98)
03:41:36.257449 IP 192.168.92.128.58494 > 192.168.92.2.domain: 58142+ PTR? 128.92.168.192.in-addr.arpa. (45)
03:41:36.260393 IP 192.168.92.2.domain > 192.168.92.128.58494: 58142 NXDomain 0/1/0 (100)
03:41:36.625402 IP 192.168.92.128.36382 > ntp7.mum-in.hosts.301-moved.de.ntp: NTPv4, Client, length 48
03:41:36.667995 IP ntp7.mum-in.hosts.301-moved.de.ntp > 192.168.92.128.36382: NTPv4, Server, length 48
03:41:36.670959 IP 192.168.92.128.45630 > 192.168.92.2.domain: 26376+ PTR? 39.210.46.192.in-addr.arpa. (44)
```

If you can't understand the output right now, you first need to get familiar with the tcpdump output format.

Check available network interfaces with tcpdump

By default, tcpdump captures traffic from any available network interface. If there are multiple network interfaces active, you may want to specify the network interface from which tcpdump will capture packets. To start tcpdump on a specific interface, you will first have to find out the interface name.

Here's how to list all available network interfaces with tcpdump:

```
sudo tcpdump -D
```

Alternatively, you can add the **--list-interface** flag to the command:

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

```
File Actions Edit View Help
kali@kali -
(kali@kali)-[~]
└─$ sudo tcpdump -D
1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]
(kali@kali)-[~]
└─$
```

The returned output contains a list of all active network interfaces that tcpdump can listen on. To configure tcpdump to capture transmissions from a specific network interface, enter the following command:

```
sudo tcpdump -i interface_id
```

Alternatively, you can add the **--interface** flag to the command:

```
sudo tcpdump --interface interface_id
```

Now that you have captured a few packets, let's take a closer look at them and see how you can tweak the output to make it more readable.

Explore tcpdump filters

Tcpdump has the potential to capture a large amount of traffic in a single run. Such an information overload can throw you off track when investigating or troubleshooting a problem with a particular host or network protocol.

This is where tcpdump filters come in handy. You can chain the tcpdump command with certain flags to filter out network traffic and capture specific packets. You can then store these packets and analyze them later to find the root of any network-related issues. Let's take a look at how to use filters in tcpdump.

Packet filtering based on the network protocol being used

To filter packets transmitted over a specific protocol, enter the protocol name using the tcpdump command and it will capture only packets transmitted over the specified network protocol.

For example, to capture ICMP-based packets, you simply append icmp to the end of the tcpdump command. The process is the same if you only want to capture UDP or TCP packets.

```
sudo tcpdump -c 5 icmp
```

This command will only return output if there is data exchange via the ICMP protocol.

```
kali@kali:~$ sudo tcpdump -c 5 icmp
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:26:55.809302 IP 192.168.92.128 > del03s09-in-f4.1e100.net: ICMP echo request, id 1398, seq 1, length 64
17:26:56.842046 IP del03s09-in-f4.1e100.net > 192.168.92.128: ICMP echo reply, id 1398, seq 1, length 64
17:26:56.811451 IP 192.168.92.128 > del03s09-in-f4.1e100.net: ICMP echo request, id 1398, seq 2, length 64
17:26:56.845044 IP del03s09-in-f4.1e100.net > 192.168.92.128: ICMP echo reply, id 1398, seq 2, length 64
17:27:02.663853 IP 192.168.92.128 > del03s09-in-f4.1e100.net: ICMP echo request, id 46812, seq 1, length 64
5 packets captured
6 packets received by filter
0 packets dropped by kernel
kali@kali:~$
```

Host-based packet filtering

You can configure tcpdump to capture packets related to a single host with the host parameter. This is especially useful when all systems on your network are up and running except for one. This filter allows you to perform targeted investigations and speeds up the overall troubleshooting process because you are not distracted by unnecessary data.

To capture packets related to a specific host, specify the host's network address with the host parameter:

```
sudo tcpdump -c 5 host 192.168.2.1
```

Similar to the network protocol filter, this command will only return output if any ongoing transmissions are related to the specified host.

```
kali@kali:~$ sudo tcpdump -c 5 host 192.168.92.128
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:38:25.900243 IP 192.168.92.128.49939 > 192.168.92.2.domain: 56783+ A? content-signature-2.cdn.mozilla.net. (53)
17:38:25.900318 IP 192.168.92.128.49939 > 192.168.92.2.domain: 54477+ AAAA? content-signature-2.cdn.mozilla.net. (53)
17:38:25.928230 IP 192.168.92.2.domain > 192.168.92.128.49939: 54477 9/4/3 CNAME d2nxq2uap88usk.cloudfront.net., AAAA 2600:9000:2041:8600:a:da5e:7900:93a1, AAAA 2600:9000:2041:ce00:a:da5e:7900:93a1, AAAA 2600:9000:2041:a800:a:da5e:7900:93a1, AAAA 2600:9000:2041:3e00:a:da5e:7900:93a1, AAAA 2600:9000:2041:3000:a:da5e:7900:93a1, AAAA 2600:9000:2041:6000:a:da5e:7900:93a1, AAAA 2600:9000:2041:0:a:da5e:7900:93a1, AAAA 2600:9000:2041:ba00:a:da5e:7900:93a1 (507)
17:38:25.928231 IP 192.168.92.2.domain > 192.168.92.128.49939: 56783 5/4/8 CNAME d2nxq2uap88usk.cloudfront.net., A 52.85.234.7, A 52.85.234.14, A 52.85.234.114, A 52.85.234.6 (470)
17:38:26.004220 IP 192.168.92.128.55869 > 192.168.92.2.domain: 61503+ PTR? 2.92.168.192.in-addr.arpa. (43)
5 packets captured
8 packets received by filter
0 packets dropped by kernel
kali@kali:~$
```

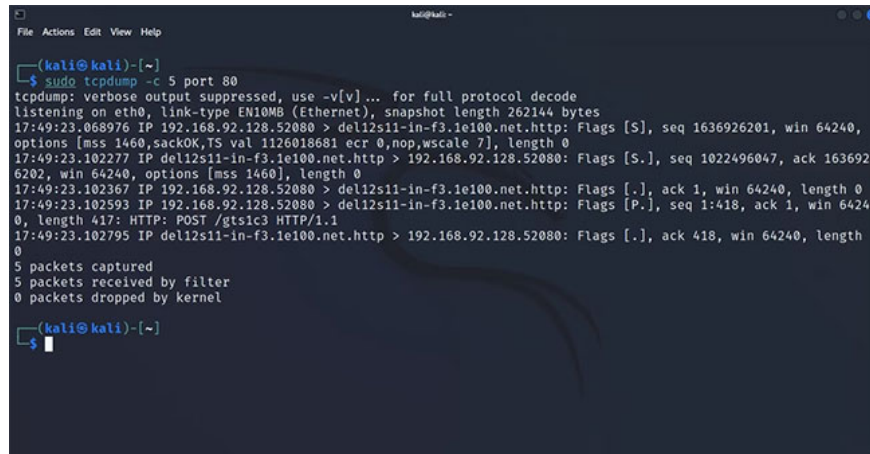
Active Port Based Packet Filtering

Tcpdump comes equipped with a parameter that allows you to filter network traffic and capture only packets transmitted to or from a specific port.

To capture packets coming from a specific port, append the port flag to the tcpdump command and specify the port number next to it. For example, to capture all incoming or outgoing HTTP traffic, specify port 80:

```
sudo tcpdump -c 5 port 80
```

Tcpdump will listen on port 80, waiting for HTTP transmissions. When it detects HTTP packets in the network, it will capture them.



```
(kali@kali)-[~]
└─$ sudo tcpdump -c 5 port 80
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:49:23.068976 IP 192.168.92.128.52080 > del12s11-in-f3.1e100.net.http: Flags [S], seq 1636926201, win 64240,
options [mss 1460,sackOK,TS val 1126018681 ecr 0,nop,wscale 7], length 0
17:49:23.102277 IP del12s11-in-f3.1e100.net.http > 192.168.92.128.52080: Flags [S.], seq 1022496047, ack 163692
6202, win 64240, options [mss 1460], length 0
17:49:23.102367 IP 192.168.92.128.52080 > del12s11-in-f3.1e100.net.http: Flags [..], ack 1, win 64240, length 0
17:49:23.102593 IP 192.168.92.128.52080 > del12s11-in-f3.1e100.net.http: Flags [P.], seq 1:418, ack 1, win 6424
0, length 417: HTTP: POST /gts1c3 HTTP/1.1
17:49:23.102795 IP del12s11-in-f3.1e100.net.http > 192.168.92.128.52080: Flags [..], ack 418, win 64240, length
0
5 packets captured
5 packets received by filter
0 packets dropped by kernel

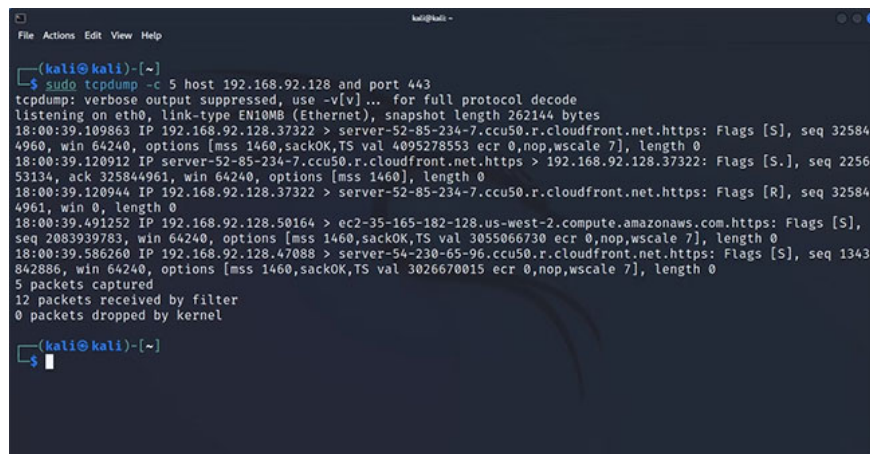
(kali@kali)-[~]
└─$
```

Combine filters together for advanced sorting

The previous sections discussed how you can filter traffic based on port, protocol, or host, but what if you want to capture traffic from a specific host's port using a specific network protocol? Well, you're in luck because this is possible, thanks to the ability to use logical operators with the tcpdump command.

To capture packets from an individual host using port 443, use the following command:

```
sudo tcpdump -c 5 host 192.168.2.1 and port 443
```



```
(kali@kali)-[~]
└─$ sudo tcpdump -c 5 host 192.168.2.1 and port 443
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:00:39.109863 IP 192.168.92.128.37322 > server-52-85-234-7.ccu50.r.cloudfront.net.https: Flags [S], seq 32584
4960, win 64240, options [mss 1460,sackOK,TS val 4095278553 ecr 0,nop,wscale 7], length 0
18:00:39.120912 IP server-52-85-234-7.ccu50.r.cloudfront.net.https > 192.168.92.128.37322: Flags [S.], seq 2256
53134, ack 325844961, win 64240, options [mss 1460], length 0
18:00:39.120944 IP 192.168.92.128.37322 > server-52-85-234-7.ccu50.r.cloudfront.net.https: Flags [R], seq 32584
4961, win 0, length 0
18:00:39.491252 IP 192.168.92.128.50164 > ec2-35-165-182-128.us-west-2.compute.amazonaws.com.https: Flags [S],
seq 2083939783, win 64240, options [mss 1460,sackOK,TS val 3055066730 ecr 0,nop,wscale 7], length 0
18:00:39.586260 IP 192.168.92.128.47088 > server-94-230-65-96.ccu50.r.cloudfront.net.https: Flags [S], seq 1343
842886, win 64240, options [mss 1460,sackOK,TS val 3026670015 ecr 0,nop,wscale 7], length 0
5 packets captured
12 packets received by filter
0 packets dropped by kernel

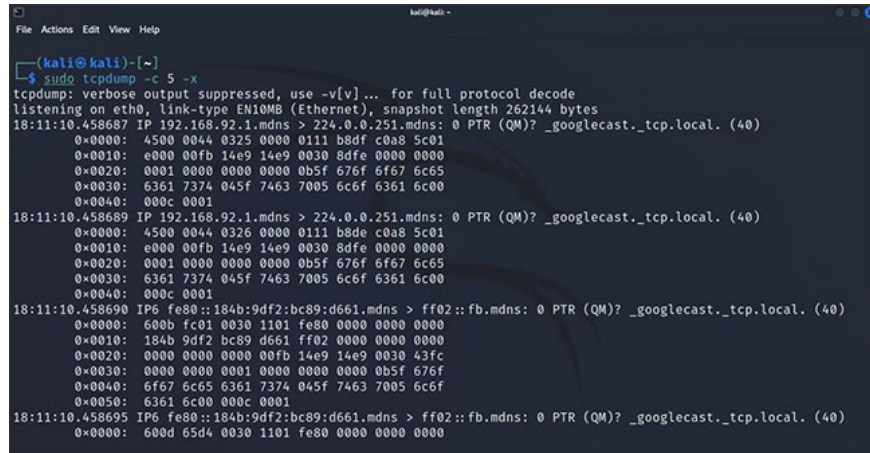
(kali@kali)-[~]
└─$
```

Check the contents of captured packets

By default, tcpdump displays the headers of a packet in its output. While this is more than enough in most cases, sometimes you may want or need to take a deeper look at the captured data. You can pass certain parameters to the tcpdump command to examine the contents of the captured packet.

Here's how to view the contents of the packages:

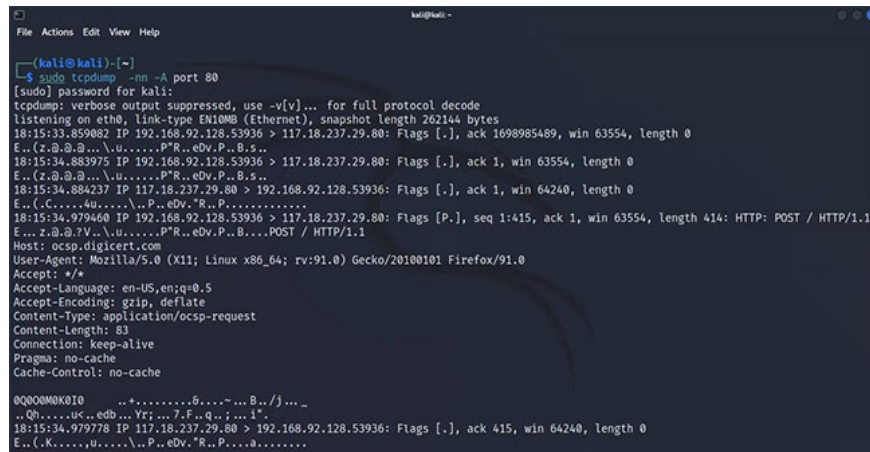
```
sudo tcpdump -c 5 -x
```



```
(kali@kali)-[~]
└─$ sudo tcpdump -c 5 -x
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:11:10.458687 IP 192.168.92.1.mdns > 224.0.0.251.mdns: 0 PTR (QM)? _googlecast._tcp.local. (40)
0x0000: 4500 0044 0325 0000 0111 b8df c0a8 5c01
0x0010: e000 00fb 14e9 14e9 0030 8dfe 0000 0000
0x0020: 0001 0000 0000 0000 0b5f 676f 6f67 6c65
0x0030: 6361 7374 045f 7463 7005 6c6f 6361 6c00
0x0040: 000c 0001
18:11:10.458689 IP 192.168.92.1.mdns > 224.0.0.251.mdns: 0 PTR (QM)? _googlecast._tcp.local. (40)
0x0000: 4500 0044 0326 0000 0111 b8de c0a8 5c01
0x0010: e000 00fb 14e9 14e9 0030 8dfe 0000 0000
0x0020: 0001 0000 0000 0000 0b5f 676f 6f67 6c65
0x0030: 6361 7374 045f 7463 7005 6c6f 6361 6c00
0x0040: 000c 0001
18:11:10.458690 IP6 fe80::184b:9df2:bc89:d661.mdns > ff02::fb.mdns: 0 PTR (QM)? _googlecast._tcp.local. (40)
0x0000: 600b fc01 0030 1101 fe80 0000 0000 0000
0x0010: 184b 9df2 bc89 d661 ff02 0000 0000 0000
0x0020: 0000 0000 0000 00fb 14e9 14e9 0030 43fc
0x0030: 0000 0000 0001 0000 0000 0000 0b5f 676f
0x0040: 6f67 6c65 6361 7374 045f 7463 7005 6c6f
0x0050: 6361 6c00 000c 0001
18:11:10.458695 IP6 fe80::184b:9df2:bc89:d661.mdns > ff02::fb.mdns: 0 PTR (QM)? _googlecast._tcp.local. (40)
0x0000: 600d 65d4 0030 1101 fe80 0000 0000 0000
```

This command returns the hex version of the contents of a captured packet. If you want to see the ASCII form of the data, you can pass the **-A** parameter with:

```
sudo tcpdump -A
```



```
(kali@kali)-[~]
└─$ sudo tcpdump -nn -A port 80
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:15:33.859882 IP 192.168.92.128.53936 > 117.18.237.29.80: Flags [.], ack 1698985489, win 63554, length 0
E..(z.@.@.@.\u.....P.R..eDv.P..B.s..
18:15:34.883975 IP 192.168.92.128.53936 > 117.18.237.29.80: Flags [.], ack 1, win 63554, length 0
E..(z.@.@.@.\u.....P.R..eDv.P..B.s..
18:15:34.884237 IP 117.18.237.29.80 > 192.168.92.128.53936: Flags [.], ack 1, win 64240, length 0
E..(C.....4u.....\P..eDv."R..P.....
18:15:34.979460 IP 192.168.92.128.53936 > 117.18.237.29.80: Flags [P.], seq 1:415, ack 1, win 63554, length 414: HTTP: POST / HTTP/1.1
E...z.@.@.7V..\u.....P.R..eDv.P..B....POST / HTTP/1.1
Host: ocsip.digicart.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/ocsp-request
Content-Length: 83
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

0000000010 ..+.....6....~...B../j..._
..Qh....uc..edb...Yr;...7.F..q.;...1".
18:15:34.979778 IP 117.18.237.29.80 > 192.168.92.128.53936: Flags [.], ack 415, win 64240, length 0
E..(K.....u.....\P..eDv."R..P....a.....
```

Save tcpdump output to a file

Like most other Linux command line tools, you can save the output generated by tcpdump to a file for later reference.

This can be done by adding the **-w** flag to the command. Once executed, tcpdump will store the captured data in a **.pcap** file for later analysis using tcpdump or other network monitoring tools like Wireshark.

Enter this command to save your tcpdump command output to a file:

```
sudo tcpdump -w capture.pcap
```

To read the **.pcap** file, you can use tcpdump with the **-r** parameter :

```
sudo tcpdump -r capture.pcap
```

Linux offers a plethora of networking tools that can solve any networking problem as long as it is on the software side. Knowing how to use some of the best networking tools in Linux will definitely come in handy, whether you are a sysadmin who manages networks for a living or just a daily Linux user.

Since the actual list of available network commands can be overwhelming, here is a list of some of the most important Linux networking tools you should know.

You finished reading the article "**Using tcpdump to analyze traffic**" edited by the [TipsMake](#) team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.