10 year anniversary

LOG IN

Main menu

[Articles](#) [Resources](#) [Downloads](#) [About](#)

[Open Organization](#)

An introduction to using tcpdump at the Linux command line

This flexible, powerful command-line tool helps ease the pain of troubleshooting network issues.

01 Sep 2020 | Ricardo Gerardi (Red Hat) (/users/rgerardi) | 223 | 5 comments

Image credits :
iradaturrahmat (https://pixabay.com/en/users/iradaturrahmat-3964359/)via Pixabay (https://pixabay.com/en/ubuntu-computer-program-interface-3145957/), CC0

In my experience as a sysadmin, I have often found network connectivity issues challenging to troubleshoot. For those situations, tcpdump is a great ally.

The Linux Terminal

- Top 7 terminal emulators for Linux (https://opensource.com/life/17/10/top-terminal-emulators?intcmp=7016000000127cYAAQ)
- 10 command-line tools for data analysis in Linux (https://opensource.com/article/17/2/command-line-tools-data-analysis-linux?intcmp=7016000000127cYAAQ)
- Download Now: SSH cheat sheet (https://opensource.com/downloads

/cheat-sheets/advanced-linux-commands
/?intcmp=7016000000127cYAAQ)

- Linux command line tutorials (https://opensource.com/tags/command-line?intcmp=7016000000127cYAAQ)

Tcpdump is a command line utility that allows you to capture and analyze network traffic going through your system. It is often used to help troubleshoot network issues, as well as a security tool.

A powerful and versatile tool that includes many options and filters, tcpdump can be used in a variety of cases. Since it's a command line tool, it is ideal to run in remote servers or devices for which a GUI is not available, to collect data that can be analyzed later. It can also be launched in the background or as a scheduled job using tools like cron.

In this article, we'll look at some of tcpdump's most common features.

1. Installation on Linux

Tcpdump is included with several Linux distributions, so chances are, you already have it installed. Check whether tcpdump is installed on your system with the following command:

```
$ which tcpdump
/usr/sbin/tcpdump
```

If tcpdump is not installed, you can install it but using your distribution's package manager. For example, on CentOS or Red Hat Enterprise Linux, Tcpdump requires **libpcap**, which is a library for network packet capture. If it's not installed, it will be automatically added as a dependency.

You're ready to start capturing some packets.

2. Capturing packets with tcpdump

To capture packets for troubleshooting or analysis, tcpdump requires elevated permissions, so in the following examples most commands are prefixed with **sudo**.

To begin, use the command **tcpdump --list-interfaces** (or **-D** for short) to see which interfaces are available for capture:

```
$ sudo tcpdump -D
1.eth0
2.virbr0
3.eth1
4.any (Pseudo-device that captures on all interfaces)
5.lo [Loopback]
```

In the example above, you can see all the interfaces available in my machine. The special interface **any** allows capturing in any active interface.

Let's use it to start capturing some packets. Capture all packets in any interface by running this command:

```
$ sudo tcpdump --interface any
tcpdump: verbose output suppressed, use -v or -vv for full protocol de
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262
```

```
09:56:18.312482 IP rhel75.49685 > gateway.domain: 34242+ PTR? 28.64.1
09:56:18.322425 IP gateway.domain > rhel75.49685: 34242 NXDomain* 0/1
09:56:18.323164 IP rhel75.56631 > gateway.domain: 29904+ PTR? 1.122.1
09:56:18.323342 IP rhel75.localdomain.ssh > 192.168.64.1.56322: Flags
09:56:18.323563 IP 192.168.64.1.56322 > rhel75.localdomain.ssh: Flags
09:56:18.335569 IP gateway.domain > rhel75.56631: 29904 NXDomain* 0/1
09:56:18.336429 IP rhel75.44007 > gateway.domain: 61677+ PTR? 98.122.
09:56:18.336655 IP gateway.domain > rhel75.44007: 61677* 1/0/0 PTR rh
09:56:18.337177 IP rhel75.localdomain.ssh > 192.168.64.1.56322: Flags

---- SKIPPING LONG OUTPUT -----

09:56:19.342939 IP 192.168.64.1.56322 > rhel75.localdomain.ssh: Flags
^C
9003 packets captured
9010 packets received by filter
7 packets dropped by kernel
$
```

Tcpdump continues to capture packets until it receives an interrupt signal. You can interrupt capturing by pressing **`Ctrl+C`**. As you can see in this example, **`tcpdump`** captured more than 9,000 packets. In this case, since I am connected to this server using **`ssh`**, tcpdump captured all these packets. To limit the number of packets captured and stop **`tcpdump`**, use the **`-c`** (for *count*) option:

```
$ sudo tcpdump -i any -c 5
tcpdump: verbose output suppressed, use -v or -vv for full protocol d
listening on any, link-type LINUX_SLL (Linux cooked), capture size 26
11:21:30.242740 IP rhel75.localdomain.ssh > 192.168.64.1.56322: Flags
11:21:30.242906 IP 192.168.64.1.56322 > rhel75.localdomain.ssh: Flags
11:21:30.244442 IP rhel75.43634 > gateway.domain: 57680+ PTR? 1.64.16
11:21:30.244829 IP gateway.domain > rhel75.43634: 57680 NXDomain 0/0/
11:21:30.247048 IP rhel75.33696 > gateway.domain: 37429+ PTR? 28.64.1
5 packets captured
```

In this case, **`tcpdump`** stopped capturing automatically after capturing five packets. This is useful in different scenarios—for instance, if you're troubleshooting connectivity and capturing a few initial packets is enough. This is even more useful when we apply filters to capture specific packets (shown below).

By default, tcpdump resolves IP addresses and ports into names, as shown in the previous example. When troubleshooting network issues, it is often easier to use the IP addresses and port numbers; disable name resolution by using the option **`-n`** and port resolution with **`-nn`**:

```
$ sudo tcpdump -i any -c5 -nn
tcpdump: verbose output suppressed, use -v or -vv for full protocol de
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262
23:56:24.292206 IP 192.168.64.28.22 > 192.168.64.1.35110: Flags [P.],
23:56:24.292357 IP 192.168.64.1.35110 > 192.168.64.28.22: Flags [.], a
23:56:24.292570 IP 192.168.64.28.22 > 192.168.64.1.35110: Flags [P.],
23:56:24.292655 IP 192.168.64.1.35110 > 192.168.64.28.22: Flags [.], a
23:56:24.292752 IP 192.168.64.28.22 > 192.168.64.1.35110: Flags [P.],
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```

As shown above, the capture output now displays the IP addresses and port numbers. This also prevents tcpdump from issuing DNS lookups, which helps to lower network traffic while troubleshooting network issues.

Now that you're able to capture network packets, let's explore what this output means.

3. Understanding the output format

here, to help you get started, let's explore the TCP packet. You can find more details about the different protocol formats in tcpdump's manual pages (http://www.tcpdump.org/manpages/tcpdump.1.html#lbAG). A typical TCP packet captured by tcpdump looks like this:

```
08:41:13.729687 IP 192.168.64.28.22 > 192.168.64.1.41916: Flags [P.],
```

The fields may vary depending on the type of packet being sent, but this is the general format.

The first field, `08:41:13.729687,` represents the timestamp of the received packet as per the local clock.

Next, `IP` represents the network layer protocol—in this case, `IPv4`. For `IPv6` packets, the value is `IP6`.

The next field, `192.168.64.28.22`, is the source IP address and port. This is followed by the destination IP address and port, represented by `192.168.64.1.41916`.

After the source and destination, you can find the TCP Flags `Flags [P.]`. Typical values for this field include:

Value	Flag Type	Description
S	SYN	Connection Start
F	FIN	Connection Finish
P	PUSH	Data push

.	ACK	Acknowledgment

This field can also be a combination of these values, such as `[S.]` for a `SYN-ACK` packet.

Next is the sequence number of the data contained in the packet. For the first packet captured, this is an absolute number. Subsequent packets use a relative number to make it easier to follow. In this example, the sequence is `seq 196:568,` which means this packet contains bytes 196 to 568 of this flow.

This is followed by the Ack Number: `ack 1`. In this case, it is 1 since this is the side sending data. For the side receiving data, this field represents the next expected byte (data) on this flow. For example, the Ack number for the next packet in this flow would be 568.

The next field is the window size `win 309`, which represents the number of bytes available in the receiving buffer, followed by TCP options such as the MSS (Maximum Segment Size) or Window Scale. For details about TCP protocol options, consult Transmission Control Protocol (TCP) Parameters (https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml).

Finally, we have the packet length, `length 372`, which represents the length, in bytes, of the payload data. The length is the difference between the last and first bytes in the sequence number.

Now let's learn how to filter packets to narrow down results and make it easier to troubleshoot specific issues.

which are not even related to the issue you're troubleshooting. For example, if you're troubleshooting a connectivity issue with a web server you're not interested in the SSH traffic, so removing the SSH packets from the output makes it easier to work on the real issue.

One of tcpdump's most powerful features is its ability to filter the captured packets using a variety of parameters, such as source and destination IP addresses, ports, protocols, etc. Let's look at some of the most common ones.

Protocol

To filter packets based on protocol, specifying the protocol in the command line. For example, capture ICMP packets only by using this command:

```
$ sudo tcpdump -i any -c5 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol de
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262
```

In a different terminal, try to ping another machine:

```
$ ping opensource.com
PING opensource.com (54.204.39.132) 56(84) bytes of data.
64 bytes from ec2-54-204-39-132.compute-1.amazonaws.com (54.204.39.13
```

Back in the tcpdump capture, notice that tcpdump captures and displays only the ICMP-related packets. In this case, tcpdump is not displaying name resolution packets that were generated when resolving the name `opensource.com`:

```
09:34:21.180020 IP ec2-54-204-39-132.compute-1.amazonaws.com > rhel75
09:34:22.141777 IP rhel75 > ec2-54-204-39-132.compute-1.amazonaws.com
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Host

Limit capture to only packets related to a specific host by using the **`host`** filter:

```
$ sudo tcpdump -i any -c5 -nn host 54.204.39.132
tcpdump: verbose output suppressed, use -v or -vv for full protocol d
listening on any, link-type LINUX_SLL (Linux cooked), capture size 26
09:54:20.042023 IP 192.168.122.98.39326 > 54.204.39.132.80: Flags [S]
09:54:20.088127 IP 54.204.39.132.80 > 192.168.122.98.39326: Flags [S.
09:54:20.088204 IP 192.168.122.98.39326 > 54.204.39.132.80: Flags [.]
09:54:20.088734 IP 192.168.122.98.39326 > 54.204.39.132.80: Flags [P.
09:54:20.129733 IP 54.204.39.132.80 > 192.168.122.98.39326: Flags [.]
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

In this example, tcpdump captures and displays only packets to and from host **`54.204.39.132`**.

Port

To filter packets based on the desired service or port, use the **`port`** filter. For example, capture packets related to a web (HTTP) service by using this command:

```
09:58:28.834026 IP 54.204.39.132.80 > 192.168.122.98.39330: Flags [S.
09:58:28.834093 IP 192.168.122.98.39330 > 54.204.39.132.80: Flags [.]
09:58:28.834588 IP 192.168.122.98.39330 > 54.204.39.132.80: Flags [P.
09:58:28.878445 IP 54.204.39.132.80 > 192.168.122.98.39330: Flags [.]
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Source IP/hostname

You can also filter packets based on the source or destination IP Address or hostname. For example, to capture packets from host **`192.168.122.98`**:

```
$ sudo tcpdump -i any -c5 -nn src 192.168.122.98
tcpdump: verbose output suppressed, use -v or -vv for full protocol de
listening on any, link-type LINUX_SLL (Linux cooked), capture size 26
10:02:15.220824 IP 192.168.122.98.39436 > 192.168.122.1.53: 59332+ A?
10:02:15.220862 IP 192.168.122.98.39436 > 192.168.122.1.53: 20749+ AA
10:02:15.364062 IP 192.168.122.98.39334 > 54.204.39.132.80: Flags [S]
10:02:15.409229 IP 192.168.122.98.39334 > 54.204.39.132.80: Flags [.]
10:02:15.409667 IP 192.168.122.98.39334 > 54.204.39.132.80: Flags [P.
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Notice that tcpdumps captured packets with source IP address **`192.168.122.98`** for multiple services such as name resolution (port 53) and HTTP (port 80). The response packets are not displayed since their source IP is different.

Conversely, you can use the **`dst`** filter to filter by destination IP/hostname:

```
10:05:03.572944 IP 192.168.122.1.53 > 192.168.122.98.47049: 33770 0/0,
10:05:03.621833 IP 54.204.39.132.80 > 192.168.122.98.39338: Flags [S.
10:05:03.667767 IP 54.204.39.132.80 > 192.168.122.98.39338: Flags [.]
10:05:03.672221 IP 54.204.39.132.80 > 192.168.122.98.39338: Flags [P.
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Complex expressions

You can also combine filters by using the logical operators `and` and `or` to create more complex expressions. For example, to filter packets from source IP address `192.168.122.98` and service HTTP only, use this command:

```
$ sudo tcpdump -i any -c5 -nn src 192.168.122.98 and port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol de
listening on any, link-type LINUX_SLL (Linux cooked), capture size 26
10:08:00.472696 IP 192.168.122.98.39342 > 54.204.39.132.80: Flags [S]
10:08:00.516118 IP 192.168.122.98.39342 > 54.204.39.132.80: Flags [.]
10:08:00.516583 IP 192.168.122.98.39342 > 54.204.39.132.80: Flags [P.
10:08:00.567044 IP 192.168.122.98.39342 > 54.204.39.132.80: Flags [.]
10:08:00.788153 IP 192.168.122.98.39342 > 54.204.39.132.80: Flags [F.
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

You can create more complex expressions by grouping filter with parentheses. In this case, enclose the entire filter expression with quotation marks to prevent the shell from confusing them with shell expressions:

```
$ sudo tcpdump -i any -c5 -nn "port 80 and (src 192.168.122.98 or src
10:10:37.651097 IP 192.168.122.98.39346 > 54.204.39.132.80: Flags [P.
10:10:37.692900 IP 54.204.39.132.80 > 192.168.122.98.39346: Flags [.]
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

In this example, we're filtering packets for HTTP service only (port 80) and source IP addresses `192.168.122.98` or `54.204.39.132`. This is a quick way of examining both sides of the same flow.

5. Checking packet content

In the previous examples, we're checking only the packets' headers for information such as source, destinations, ports, etc. Sometimes this is all we need to troubleshoot network connectivity issues. Sometimes, however, we need to inspect the content of the packet to ensure that the message we're sending contains what we need or that we received the expected response. To see the packet content, tcpdump provides two additional flags: `-X` to print content in hex, and ASCII or `-A` to print the content in ASCII.

For example, inspect the HTTP content of a web request like this:

```
$ sudo tcpdump -i any -c10 -nn -A port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol de
listening on any, link-type LINUX_SLL (Linux cooked), capture size 26
13:02:14.871803 IP 192.168.122.98.39366 > 54.204.39.132.80: Flags [S]
E..<..@.@.....zb6.'....P...@......r............
............................
13:02:14.910734 IP 54.204.39.132.80 > 192.168.122.98.39366: Flags [S.
E..<..@./..a6.'...zb.P..o..&...A..q a.........
.R.W.......     ................
```

```
E.....@.@..1..zb6.'....P...Ao..'..........
.....R.WGET / HTTP/1.1
User-Agent: Wget/1.14 (linux-gnu)
Accept: */*
Host: opensource.com
Connection: Keep-Alive

................
13:02:14.951199 IP 54.204.39.132.80 > 192.168.122.98.39366: Flags [.]
E..4.F@./.."6.'...zb.P..o..'.......9.2.....
.R.a....................
13:02:14.955030 IP 54.204.39.132.80 > 192.168.122.98.39366: Flags [P.
E....G@./...6.'...zb.P..o..'.......9.......
.R.b....HTTP/1.1 302 Found
Server: nginx
Date: Sun, 23 Sep 2018 17:02:14 GMT
Content-Type: text/html; charset=iso-8859-1
Content-Length: 207
X-Content-Type-Options: nosniff
Location: https://opensource.com/
Cache-Control: max-age=1209600
Expires: Sun, 07 Oct 2018 17:02:14 GMT
X-Request-ID: v-6baa3acc-bf52-11e8-9195-22000ab8cf2d
X-Varnish: 632951979
Age: 0
Via: 1.1 varnish (Varnish/5.2)
X-Cache: MISS
Connection: keep-alive

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="https://opensource.com/">here</a>.
</body></html>
................
```

```
.....R.b................
13:02:15.236592 IP 54.204.39.132.80 > 192.168.122.98.39366: Flags [F.
E..4.H@./.. 6.'...zb.P..o.........9.I.....
.R......................
13:02:15.236656 IP 192.168.122.98.39366 > 54.204.39.132.80: Flags [.]
E..4..@.@.....zb6.'....P....o.............
.....R..................
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

This is helpful for troubleshooting issues with API calls, assuming the calls are using plain HTTP. For encrypted connections, this output is less useful.

6. Saving captures to a file

Another useful feature provided by tcpdump is the ability to save the capture to a file so you can analyze the results later. This allows you to capture packets in batch mode overnight, for example, and verify the results in the morning. It also helps when there are too many packets to analyze since real-time capture can occur too fast.

To save packets to a file instead of displaying them on screen, use the option `-w` (for *write*):

```
$ sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80
[sudo] password for ricardo:
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), captur
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

As shown in this example, nothing gets displayed on-screen, and the capture finishes after capturing 10 packets, as per the option **`-c10`**. If you want some feedback to ensure packets are being captured, use the option **`-v`**.

Tcpdump creates a file in binary format so you cannot simply open it with a text editor. To read the contents of the file, execute tcpdump with the **`-r`** (for *read*) option:

```
$ tcpdump -nn -r webserver.pcap
reading from file webserver.pcap, link-type LINUX_SLL (Linux cooked)
13:36:57.679494 IP 192.168.122.98.39378 > 54.204.39.132.80: Flags [S]
13:36:57.718932 IP 54.204.39.132.80 > 192.168.122.98.39378: Flags [S.
13:36:57.719005 IP 192.168.122.98.39378 > 54.204.39.132.80: Flags [.]
13:36:57.719186 IP 192.168.122.98.39378 > 54.204.39.132.80: Flags [P.
13:36:57.756979 IP 54.204.39.132.80 > 192.168.122.98.39378: Flags [.]
13:36:57.760122 IP 54.204.39.132.80 > 192.168.122.98.39378: Flags [P.
13:36:57.760182 IP 192.168.122.98.39378 > 54.204.39.132.80: Flags [.]
13:36:57.977602 IP 192.168.122.98.39378 > 54.204.39.132.80: Flags [F.
13:36:58.022089 IP 54.204.39.132.80 > 192.168.122.98.39378: Flags [F.
13:36:58.022132 IP 192.168.122.98.39378 > 54.204.39.132.80: Flags [.]
$
```

Since you're no longer capturing the packets directly from the network interface, **`sudo`** is not required to read the file.

You can also use any of the filters we've discussed to filter the content from the file, just as you would with real-time data. For example, inspect the packets in the capture file from source IP address **`54.204.39.132`** by executing this command:

```
13:36:57.760122 IP 54.204.39.132.80 > 192.168.122.98.39378: Flags [P.
13:36:58.022089 IP 54.204.39.132.80 > 192.168.122.98.39378: Flags [F.
```

What's next?

These basic features of tcpdump will help you get started with this powerful and versatile tool. To learn more, consult the tcpdump website (http://www.tcpdump.org/#) and man pages (http://www.tcpdump.org /manpages/tcpdump.1.html).

The tcpdump command line interface provides great flexibility for capturing and analyzing network traffic. If you need a graphical tool to understand more complex flows, look at Wireshark (https://www.wireshark.org/).

One benefit of Wireshark is that it can read `.pcap` files captured by tcpdump. You can use tcpdump to capture packets in a remote machine that does not have a GUI and analyze the result file with Wireshark, but that is a topic for another day.

This article was originally published in October 2018 and has been updated by Seth Kenlon (/users/seth).

Topics : **Linux** (/tags/linux) **Command line** (/tags/command-line)

About the author

Recommended reading

[(/users/rgerardi)](/users/rgerardi)

Ricardo Gerardi - Ricardo Gerardi is a Senior Consultant at Red Hat Canada where he specializes in IT automation with Ansible and Openshift. He has experience in the telecommunications sector, having worked as Senior Architect at TELUS, and had previous experience as Senior Consultant and Pre-Sales specialist for Network Management solutions at IBM Brazil and IBM Canada for 13 years. Ricardo has been a Linux enthusiast for over 20 years. He is currently interested in hacking stuff using the Go Programming...

[• More about me (/users/rgerardi)](/users/rgerardi)

Manage your SSH connections with this open source tool (/article/20/9/ssh-connection-manager?utm_campaign=intrel)

A beginner's guide to SSH for remote connection on Linux (/article/20/9/ssh?utm_campaign=intrel)

Program hardware from the Linux command line (/article/20/9/hardware-command-line?utm_campaign=intr

A practical guide to learning awk (/article/20/9/awk-ebook?utm_campaign=intrel)

Open ports and route traffic through your firewall (/article/20/9/firewall?utm_campaign=intrel)

Try Linux on your Mac with open source virtualization (/article/20/9/try-linux-mac?utm_campaign=int

5 Comments

Eduardo Toro on 10 Oct 2018	3

dogsleg (/users/dogsleg) on 11 Oct 2018 2

True. Simply the best post I've read there so far.

sunawang (/users/sunawang) on 11 Oct 2018 2

Nice share, Ricardo. I used use this tool before I knew Wireshark. This makes me want to make a nostalgic with Tdpdump

Psteve on 16 Oct 2018 2

My guess is it's because autocorrect, but several times you use "packages" where I think you mean "packets." Am I wrong? Very good article, that minor nit aside.

Ricardo Gerardi (/users/rgerardi) on 22 Oct 2018 1

Thank you for your feedback ! You're right. I will get it fixed.

Subscribe to our weekly newsletter

Enter your email address...

Select your country or region

Subscribe

Privacy Statement

Get the highlights in your inbox every week.

Find us:

Privacy Policy | Terms of Use | Contact | Meet the Team | Visit opensource.org