DANIELMIESSLER

A Tcpdump Tutorial With Examples

CREATED: JANUARY 4, 2004 | UPDATED: OCTOBER 24, 2018

Why `tcpdump`?

TCPDUMP (HTTPS://TCPDUMP.ORG/) is the premier network analysis tool for information security professionals, and this guide will show you how to use its functionality as well as provide you with lots of examples.

> I prefer viewing traffic in `tcpdump` vs. `wireshark` in most cases because seeing raw traffic forces you to learn and stay sharp with the protocols you're dealing with.

Basics

`tcpdump` has *a ton* of options that you want to be somewhat familiar with. But you don't have to learn them all now if you're in a rush. If you're looking to jump in, just head down to BASIC COMMUNICATION.

Options

- `-i any` : Listen on all interfaces just to see if you're seeing any traffic.
- `-i eth0` : Listen on the eth0 interface.
- `-D` : Show the list of available interfaces
- `-l` : Line-readable output (for viewing as you save, or sending to other commands)
- `-A` : Display output in ASCII.
- `-n` : Don't resolve hostnames.
- `-nn` : Don't resolve hostnames *or* port names.
- `-q` : Be less verbose (more quiet) with your output.
- `-t` : Give human-readable timestamp output.
- `-tttt` : Give maximally human-readable timestamp output.
- `-X` : Show the packet's *contents* in both HEX (HTTPS://EN.WIKIPEDIA.ORG/WIKI/HEXIDECIMAL) and ASCII (HTTPS://EN.WIKIPEDIA.ORG/WIKI/ASCII).
- `-XX` : Same as `-X`, but also shows the ethernet header.
- `-v, -vv, -vvv` : Increase the amount of packet information you get back.
- `-c` : Only get *x* number of packets and then stop.
- `-s` : Define the *snaplength* (size) of the capture in bytes. Use `-s0` to get everything, unless you are intentionally capturing less.
- `-S` : Print absolute sequence numbers.
- `-e` : Get the ethernet header as well.
- `-q` : Show less protocol information.
- `-E` : Decrypt IPSEC traffic by providing an encryption key.

> The default snaplength as of `tcpdump` 4.0 has changed from 68 bytes to 96 bytes. While this will give you more of a packet to see, it still won't get everything. Use `-s 1514` or `-s 0` to get full coverage.

Expressions

In `tcpdump`, *Expressions* allow you to trim out various types of traffic and find exactly what you're looking for. Mastering the expressions and learning to combine them creatively is what makes one truly powerful with `tcpdump`.

There are three main types of expression: `type`, `dir`, and `proto`.

- Type options are: `host`, `net`, and `port`.
- Direction lets you do `src`, `dst`, and combinations thereof.
- Proto(col) lets you designate: `tcp`, `udp`, `icmp`, `ah`, and many more.

Examples

So, now that we've seen what our options are, let's look at some real-world examples that we're likely to see in our everyday work.

BASIC COMMUNICATION

Just see what's going on, by looking at all interfaces.

```
# tcpdump -i any
```

SPECIFIC INTERFACE

Basic view of what's happening on a particular interface.

```
# tcpdump -i eth0
```

RAW OUTPUT VIEW

Verbose output, with no resolution of hostnames or port numbers, absolute sequence numbers, and human-readable timestamps.

```
# tcpdump -ttttnnvvS
```

FIND TRAFFIC BY IP

One of the most common queries, this will show you traffic from 1.2.3.4, whether it's the source or the destination.

```
# tcpdump host 1.2.3.4
```

SEEING MORE OF THE PACKET WITH HEX OUTPUT

Hex output is useful when you want to see the content of the packets in question, and it's often best used when you're isolating a few candidates for closer scrutiny.

```
# tcpdump -nnvXSs 0 -c1 icmp
```

FILTERING BY SOURCE AND DESTINATION

It's quite easy to isolate traffic based on either source or destination using `src` and `dst`.

```
# tcpdump src 2.3.4.5
# tcpdump dst 3.4.5.6
```

FINDING PACKETS BY NETWORK

To find packets going to or from a particular network, use the `net` option. You can combine this with the `src` or `dst` options as well.

```
# tcpdump net 1.2.3.0/24
```

SHOW TRAFFIC RELATED TO A SPECIFIC PORT

You can find specific port traffic by using the `port` option followed by the port number.

```
# tcpdump port 3389
# tcpdump src port 1025
```

SHOW TRAFFIC OF ONE PROTOCOL

If you're looking for one particular kind of traffic, you can use tcp, udp, icmp, and many others as well.

```
# tcpdump icmp
```

SHOW ONLY IP6 TRAFFIC

You can also find all IP6 traffic using the protocol option.

```
# tcpdump ip6
```

FIND TRAFFIC USING PORT RANGES

You can also use a range of ports to find traffic.

```
# tcpdump portrange 21-23
```

FIND TRAFFIC BASED ON PACKET SIZE

If you're looking for packets of a particular size you can use these options. You can use less, greater, or their associated symbols that you would expect from mathematics.

```
# tcpdump less 32
# tcpdump greater 64
# tcpdump <= 128
```

READING / WRITING CAPTURES TO A FILE

It's often useful to save packet captures into a file for analysis in the future. These files are known as PCAP (PEE-cap) files, and they can be processed by hundreds of different applications, including network analyzers, intrusion detection systems, and of course by `tcpdump` itself. Here we're writing to a file called *capture_file* using the `-w` switch.

```
# tcpdump port 80 -w capture_file
```

You can read PCAP files by using the `-r` switch. Note that you can use all the regular commands within tcpdump while reading in a file; you're only limited by the fact that you can't capture and process what doesn't exist in the file already.

```
# tcpdump -r capture_file
```

Advanced

Now that we've seen what we can do with the basics through some examples, let's look at some more advanced stuff.

It's All About the Combinations

Being able to do these various things individually is powerful, but the real magic of `tcpdump` comes from the ability to **combine options in creative ways** in order to isolate exactly what you're looking for. There are three ways to do combinations, and if you've studied programming at all they'll be pretty familiar to you.

1. **AND**
   and or &&
2. **OR**
   or or ||
3. **EXCEPT**
   not or !

Here are some examples of combined commands.

FROM SPECIFIC IP AND DESTINED FOR A SPECIFIC PORT

Let's find all traffic from 10.5.2.3 going to any host on port 3389.

```
tcpdump -nnvvS src 10.5.2.3 and dst port 3389
```

FROM ONE NETWORK TO ANOTHER

Let's look for all traffic coming from 192.168.x.x and going to the 10.x or 172.16.x.x networks, and we're showing hex output with no hostname resolution and one level of extra verbosity.

```
tcpdump -nvX src net 192.168.0.0/16 and dst net
10.0.0.0/8 or 172.16.0.0/16
```

NON ICMP TRAFFIC GOING TO A SPECIFIC IP

This will show us all traffic going to 192.168.0.2 that is *not* ICMP.

```
tcpdump dst 192.168.0.2 and src net and not icmp
```

TRAFFIC FROM A HOST THAT ISN'T ON A SPECIFIC PORT

This will show us all traffic from a host that isn't SSH traffic (assuming default port usage).

```
tcpdump -vv src mars and not dst port 22
```

As you can see, you can build queries to find just about anything you need. The key is to first figure out *precisely* what you're looking for and then to build the syntax to isolate that specific type of traffic.

Keep in mind that when you're building complex queries you might have to group your options using single quotes. Single quotes are used in order to tell `tcpdump` to ignore certain special characters—in this case below the "( )" brackets. This same technique can be used to group using other expressions such as `host`, `port`, `net`, etc.

```
# tcpdump 'src 10.0.2.4 and (dst port 3389 or 22)'
```

ISOLATE TCP FLAGS

You can also use filters to isolate packets with specific TCP flags set.

Isolate TCP RST flags.

```
The filters below find these various packets because tcp[13] looks at offset
13 in the TCP HEADER
(HTTP://WWW.NETWORKSORCERY.COM/ENP/PROTOCOL/TCP.HTM), the number represents the location within the byte, and the !=0 means that the flag in question is set to 1, i.e. it's on.
```

```
# tcpdump 'tcp[13] & 4!=0'
# tcpdump 'tcp[tcpflags] == tcp-rst'
```

Isolate TCP SYN flags.

```
# tcpdump 'tcp[13] & 2!=0'
# tcpdump 'tcp[tcpflags] == tcp-syn'
```

Isolate packets that have both the SYN and ACK flags set.

```
# tcpdump 'tcp[13]=18'
```

> Only the PSH, RST, SYN, and FIN flags are displayed in `tcpdump`'s flag field output. URGs and ACKs are displayed, but they are shown elsewhere in the output rather than in the flags field.

Isolate TCP URG flags.

```
# tcpdump 'tcp[13] & 32!=0'
# tcpdump 'tcp[tcpflags] == tcp-urg'
```

Isolate TCP ACK flags.

```
# tcpdump 'tcp[13] & 16!=0'
# tcpdump 'tcp[tcpflags] == tcp-ack'
```

Isolate TCP PSH flags.

```
# tcpdump 'tcp[13] & 8!=0'
# tcpdump 'tcp[tcpflags] == tcp-psh'
```

Isolate TCP FIN flags.

```
# tcpdump 'tcp[13] & 1!=0'
# tcpdump 'tcp[tcpflags] == tcp-fin'
```

Everyday Recipe Examples

> Because tcpdump can output content in ASCII, you can use it to search for cleartext content using other command-line tools like `grep`.

Finally, now that we the theory out of the way, here are a number of quick recipes you can use for catching various kinds of traffic.

BOTH SYN AND RST SET

```
# tcpdump 'tcp[13] = 6'
```

FIND HTTP USER AGENTS

> The `-l` switch lets you see the traffic as you're capturing it, and helps when sending to commands like `grep`.

```
# tcpdump -vvAls0 | grep 'User-Agent:'
```

CLEARTEXT GET REQUESTS

```
# tcpdump -vvAls0 | grep 'GET'
```

FIND HTTP HOST HEADERS

```
# tcpdump -vvAls0 | grep 'Host:'
```

FIND HTTP COOKIES

```
# tcpdump -vvAls0 | grep 'Set-Cookie|Host:|Cookie:'
```

FIND SSH CONNECTIONS

This one works regardless of what port the connection comes in on, because it's getting the banner response.

```
# tcpdump 'tcp[(tcp[12]>>2):4] = 0x5353482D'
```

FIND DNS TRAFFIC

```
# tcpdump -vvAs0 port 53
```

FIND FTP TRAFFIC

```
# tcpdump -vvAs0 port ftp or ftp-data
```

FIND NTP TRAFFIC

```
# tcpdump -vvAs0 port 123
```

FIND CLEARTEXT PASSWORDS

```
# tcpdump port http or port ftp or port smtp or port
imap or port pop3 or port telnet -lA | egrep -i -B5
'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|passwd=|password=|pass:|user:|usernam
|user '
```

FIND TRAFFIC WITH EVIL BIT

There's a bit in the IP header that never gets set by legitimate applications, which we call the "Evil Bit". Here's a fun filter to find packets where it's been toggled.

```
# tcpdump 'ip[6] & 128 != 0'
```

> Check out MY OTHER TUTORIALS (HTTPS://DANIELMIESSLER.COM/STUDY/) as well.

Summary

Here are the takeaways.

1. `tcpdump` is a valuable tool for anyone looking to get into networking or information security.
2. The raw way it interfaces with traffic, combined with the precision it offers in inspecting packets make it the best possible tool for learning TCP/IP.

3. Protocol Analyzers like Wireshark are great, but if you want to truly master packet-fu, you must become one with `tcpdump` first.

Well, this primer should get you going strong, but THE MAN PAGE (HTTP://WWW.TCPDUMP.ORG/TCPDUMP_MAN.HTML) should always be handy for the most advanced and one-off usage scenarios. I truly hope this has been useful to you, and feel free to CONTACT ME (HTTPS://DANIELMIESSLER.COM/ABOUT/) if you have any questions.

Notes

1. I'm currently (sort of) writing a book on tcpdump for No Starch Press.
2. The leading image is from SECURITYWIZARDRY.COM (HTTP://SECURITYWIZARDRY.COM).
3. Some of the isolation filters borrowed from SÉBASTIEN WAINS (HTTPS://TWITTER.COM/SEBASTIENWAINS).
4. Thanks to Peter at hackertarget.com for inspiration on the new table of contents (simplified), and also for some additional higher-level protocol filters added in July 2018.
5. An anagram for the TCP flags is: **U**NSKILLED **A**TTACKERS **P**ESTER **R**EAL **S**ECURITY **F**OLK. (HTTPS://DANIELMIESSLER.COM/STUDY/TCPFLAGS/)