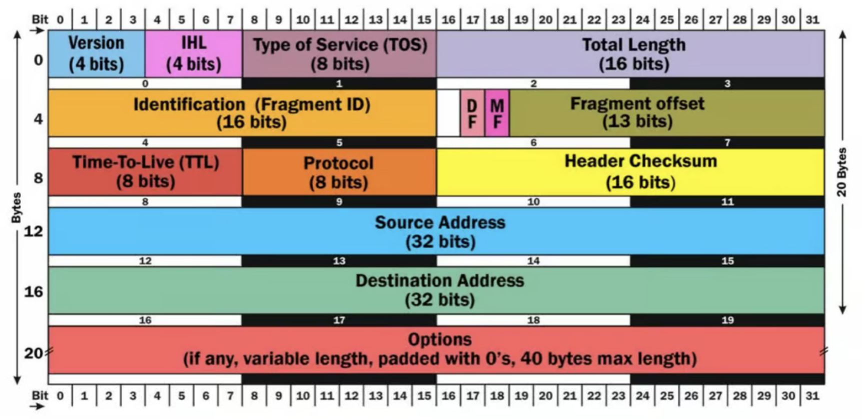
A tcpdump Tutorial with Examples

50 Ways to Isolate Traffic


Bit 0–31
Version (4 bits)
IHL (4 bits)
Type of Service (TOS) (8 bits)
Total Length (16 bits)
Identification (Fragment ID) (16 bits)
D F
M F
Fragment offset (13 bits)
Time-To-Live (TTL) (8 bits)
Protocol (8 bits)
Header Checksum (16 bits)
Source Address (32 bits)
Destination Address (32 bits)
Options (if any, variable length, padded with 0's, 40 bytes max length)
Bytes
20 Bytes

An IP Header

tcpdump is the world's premier network analysis tool—combining both power and simplicity into a single command-line interface.

This guide will show you how to isolate traffic in *multiple* ways—including by IP, port, protocol, or application **to help you find what you're looking for**.

Traffic isolation examples

Let's start with a basic command that will get us HTTPS traffic:

```
tcpdump -nnSX port 443
```

```
04:45:40.573686 IP 78.149.209.110.27782 > 172.30.0.144.443: Flags [.], ack
278239097, win 28, options [nop,nop,TS val 939752277 ecr 1208058112], length 0
    0x0000:  4500 0034 0014 0000 2e06 c005 4e8e d16e  E..4........N..n
    0x0010:  ac1e 0090 6c86 01bb 8e0a b73e 1095 9779  ....l......>...y
    0x0020:  8010 001c d202 0000 0101 080a 3803 7b55  ............8.{U
    0x0030:  4801 8100
```

You can get a single packet with `-c 1`, or `n` number with `-c n`.

This showed some HTTPS traffic, with a hex display visible on the right portion of the output (alas, it's encrypted). Just remember—when in doubt, run the command above with the port you're interested in, and you should be on your way.

Examples

PacketWizard™ isn't really trademarked, but it should be.

Now that you are able to get basic traffic, let's step through numerous examples that you are likely to need during your job in networking, security, or as any type of PacketWizard™.

Everything on an interface

Just see what's going on, by looking at what's hitting your interface.

Or get *all* interfaces with -i any.

```
tcpdump -i eth0
```

Find Traffic by IP

One of the most common queries, using host, you can see traffic that’s going to or from 1.1.1.1.

Expression Types: host, net, and port.Directions: src and dst.Types:host, net, and port. Protocols:tcp, udp, icmp, and many more.

```
tcpdump host 1.1.1.1
```

```
06:20:25.593207 IP 172.30.0.144.39270 > one.one.one.one.domain:
12790+ A? google.com.
(28) 06:20:25.594510 IP one.one.one.one.domain > 172.30.0.144.39270:
12790 1/0/0 A 172.217.15.78 (44)
```

Filtering by Source and/or Destination

If you only want to see traffic in one direction or the other, you can use src and dst.

```
tcpdump src 1.1.1.1tcpdump dst 1.0.0.1
```

Finding Packets by Network

To find packets going to or from a particular network or subnet, use the net option.

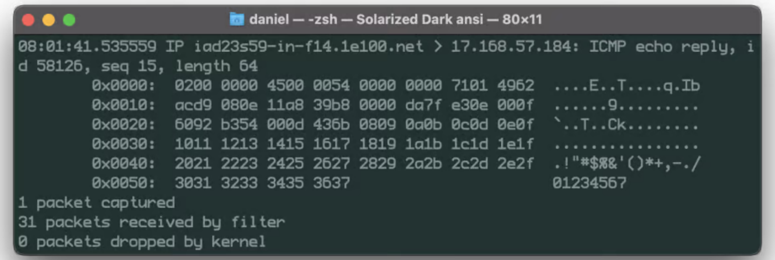
You can combine this with the src and dst options as well.

```
tcpdump net 1.2.3.0/24
```

Get Packet Contents 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 -c 1 -X icmp
```



```
08:01:41.535559 IP iad23s59-in-f14.1e100.net > 17.168.57.184: ICMP echo reply, i
d 58126, seq 15, length 64
        0x0000:  0200 0000 4500 0054 0000 0000 7101 4962  ....E..T....q.Ib
        0x0010:  acd9 080e 11a8 39b8 0000 da7f e30e 000f  ......9.........
        0x0020:  6092 b354 000d 436b 0809 0a0b 0c0d 0e0f  `..T..Ck........
        0x0030:  1011 1213 1415 1617 1819 1a1b 1c1d 1e1f  ................
        0x0040:  2021 2223 2425 2627 2829 2a2b 2c2d 2e2f  .!"#$%&'()*+,-./
        0x0050:  3031 3233 3435 3637                      01234567
1 packet captured
31 packets received by filter
0 packets dropped by kernel
```

A single ICMP packet captured by tcpdump

tcpdump is the tool everyone should learn as their base for packet analysis.

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
```

Common Options:`-nn` : Don't resolve hostnames *or* port names.`-S` : Get the entire packet.**-X** : Get hex output.

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 'len > 32 and len < 64'
```

Reading / Writing Captures to a File (pcap)

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.

More options

Here are some additional ways to tweak how you call tcpdump.

- **-X** : Show the packet's *contents* in both **hex** and **ASCII**.
- **-XX** : Same as **-X**, but also shows the ethernet header.
- **-D** : Show the list of available interfaces
- **-l** : Line-readable output (for viewing as you save, or sending to other commands)
- **-q** : Be less verbose (more quiet) with your output.
- **-t** : Give human-readable timestamp output.
- **-tttt** : Give maximally human-readable timestamp output.
- **-i eth0** : Listen on the eth0 interface.
- **-vv** : Verbose output (more v's gives more output).
- **-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.

It's All About the Combinations

Browsemy other tutorials

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 !

Raw Output View

Use this combination to see verbose output, with no resolution of hostnames or port numbers, using absolute sequence numbers, and showing human-readable timestamps.

```
tcpdump -ttnnvvS
```

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, 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-push'
```

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:|username:|password:|login:|pa
|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'
```