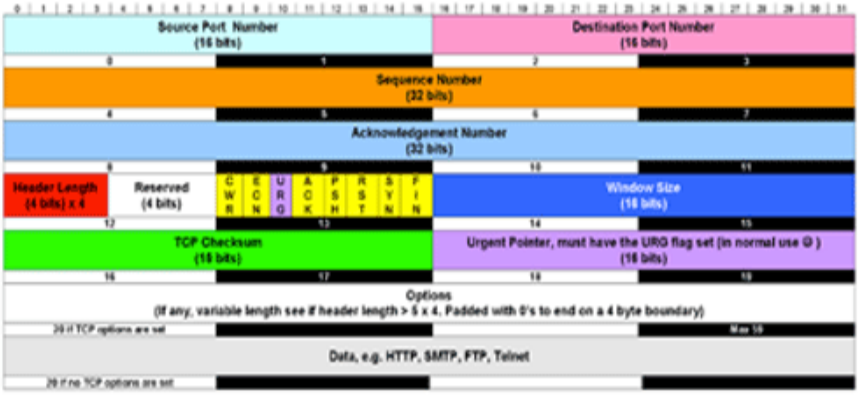
A tcpdump Tutorial and Primer with Examples



WHY `TCPDUMP`?

`tcpdump` is the premier network analysis tool for information security professionals. Having a solid grasp of this über-powerful application is mandatory for anyone desiring a thorough understanding of TCP/IP. Many prefer to use higher level analysis tools such as Wireshark, but I believe this to usually be a mistake.

When using a tool that displays network traffic a more natural (raw) way the burden of analysis is placed directly on the human rather than the application. This approach cultivates continued and elevated understanding of the TCP/IP suite, and for this reason I *strongly* advocate using `tcpdump` instead of other tools whenever possible.

```
15:31:34.079416 IP (tos 0x0, ttl 64, id 20244, offset 0, flags [DF],
proto: TCP (6), length: 60) source.35970 > dest.80: S, cksum 0x0ac1
(correct), 2647022145:2647022145(0) win 5840 0x0000: 4500 003c 4f14
4006 7417 0afb 0257  E..  0x0010: 4815 222a 8c82 0050 9dc6 5a41 0000
0000  H."*...P..ZA.... 0x0020: a002 16d0 0ac1 0000 0204 05b4
0402 080a  ..............  0x0030: 14b4 1555 0000 0000 0103 0302
```

TABLE 1. — RAW TCP/IP OUTPUT.

BASICS

Below are a few options you can use when configuring `tcpdump`. They're easy to forget and/or confuse with other types of filters, e.g., Wireshark, so hopefully this page can serve as a reference for you, as it does me. here are the main ones I like to keep in mind depending on what I'm looking at.

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
- `-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 and 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
```

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), 23:11:10.370321 IP
(tos 0x20, ttl  48, id 34859, offset 0, flags [none], length: 84)
69.254.213.43 > 72.21.34.42: icmp 64: echo request seq 0
        0x0000:  4520 0054 882b 0000 3001 7cf5 45fe d52b  E..T.+..0.|.E..+
        0x0010:  4815 222a 0800 3530 272a 0000 25ff d744  H."..50'..%..D
        0x0020:  ae5e 0500 0809 0a0b 0c0d 0e0f 1011 1213  .^..............
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
1 packets captured
1 packets received by filter
0 packets dropped by kernel
```

TABLE 2. — VERBOSE CAPTURE OF AN ICMP PACKET.

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
```

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
```

READING PCAP FILES

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 familar 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.

Complex Grouping and Special Characters

Also 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. Take a look at the command below.

 # Traffic that's from 10.0.2.4 AND destined for ports 3389 or 22 ***(incorrect)***

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

If you tried to run this otherwise very useful command, you'd get an error because of the parenthesis. You can either fix this by escaping the parenthesis (putting a \ before each one), or by putting the entire command within single quotes:

 # Traffic that's from 10.0.2.4 AND destined for ports 3389 or 22 ***(correct)***

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

Isolating Specific TCP Flags

You can also capture traffic based on specific TCP flag(s).

[NOTE: 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.]

Show me all URGENT **(URG)** packets…

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

Show me all ACKNOWLEDGE **(ACK)** packets…

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

Show me all PUSH **(PSH)** packets…

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

Show me all RESET **(RST)** packets…

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

Show me all SYNCHRONIZE **(SYN)** packets…

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

Show me all FINISH **(FIN)** packets…

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

Show me all SYNCHRONIZE/ACKNOWLEDGE **(SYNACK)** packets…

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

[Note: 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.]

As with most powerful tools, however, there are multiple ways to do things. The example below shows another way to capture packets with specific TCP flags set.

```
# tcpdump 'tcp[tcpflags] == tcp-syn'
```

Capture RST Flags Using the `tcpflags` option...

```
# tcpdump 'tcp[tcpflags] == tcp-rst'
```

Capture FIN Flags Using the `tcpflags` option...

```
# tcpdump 'tcp[tcpflags] == tcp-fin'
```

[Note: The same technique can be used for the other flags as well; they have been omitted in the interest of space.]

Identifying Noteworthy Traffic

Finally, there are a few quick recipes you'll want to remember for catching specific and specialized traffic, such as malformed / likely-malicious packets.

PACKETS WITH BOTH THE RST AND SYN FLAGS SET (THIS SHOULD NEVER BE THE CASE)

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

FIND CLEARTEXT HTTP GET REQUESTS

```
# tcpdump 'tcp[32:4] = 0x47455420'
```

FIND SSH CONNECTIONS ON ANY PORT (VIA BANNER TEXT)

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

PACKETS WITH A TTL LESS THAN 10 (USUALLY INDICATES A PROBLEM OR USE OF `TRACEROUTE`)

```
# tcpdump 'ip[8] < 10'
```

PACKETS WITH THE EVIL BIT SET (HACKER TRIVIA MORE THAN ANYTHING ELSE)

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

SUMMARY

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 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 if you have any questions.