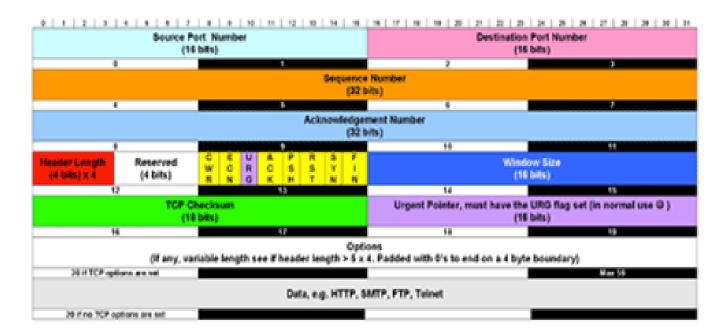- Recommended
- Home
- Blog
- Study
- Projects
- Services
- Podcast
- About

A tcpdump Primer with Examples

Home » Study » A tcpdump Primer with Examples

- Options
- Basic Usage
- Examples
- Writing to a File
- Getting Creative
- Advanced

[NOTE: For more primers like this, check out my tutorial series.]

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 ~~Ethereal~~ Wireshark, but I believe this to usually be a mistake.

In a discipline so dependent on a true understanding of concepts vs. rote learning, it's important to stay fluent in the underlying mechanics of the TCP/IP suite. A thorough grasp of these protocols allows one to troubleshoot at a level far beyond the average analyst, but mastery of the protocols is only possible through continued exposure to them.

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 4000
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
```

Options

Below are a few options (with examples) that will help you greatly when working with the tool. They're easy to forget and/or confuse with other types of filters, i.e. ethereal, so hopefully this page can serve as a reference for you, as it does me.

First off, I like to add a few options to the `tcpdump` command itself, depending on what I'm looking at. The first of these is **-n**, which requests that names are not resolved, resulting in the IPs themselves always being displayed. The second is **-X**, which displays both hex and ascii content within the packet. The final one is **-S**, which changes the display of sequence numbers to absolute rather than relative. The idea there is that you can't see weirdness in the sequence numbers if they're being hidden from you. Remember, the advantage of using `tcpdump` vs. another tool is getting manual interaction with the packets.

It's also important to note that `tcpdump` only takes the first ~~68~~ 96 bytes of data from a packet by default. If you would like to look at more, add the **-s *number*** option to the mix, where *number* is the number of bytes you want to capture. I recommend using 0 (zero) for a snaplength, which gets everything. Here's a short list of the options I use most:

[NOTE: All of the following come after `tcpdump`, for example: `tcpdump -i any`]

- **`-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.

- **-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.
- **icmp** : Only get ICMP packets.
- **-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` to get full coverage]

Basic Usage

So, based on the kind of traffic I'm looking for, I use a different combination of options to `tcpdump`, as can be seen below:

1. **Basic communication** // see the basics without many options

   # tcpdump -nS

2. **Basic communication (very verbose)** // see a good amount of traffic, with verbosity and no name help

   # tcpdump -nnvvS

3. **A deeper look at the traffic** // adds -X for payload but doesn't grab any more of the packet

   # tcpdump -nnvvXS

4. **Heavy packet viewing** // the final "s" increases the snaplength, grabbing the whole packet

   # tcpdump -nnvvXSs 1514

Here's a capture of exactly two (`-c2`) ICMP packets (a ping and pong) using some of the options described above. Notice how much we see about each packet.

```
hermes root # tcpdump -nnvXSs 0 -c2 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
```

```
23:11:10.370344 IP (tos 0x20, ttl  64, id 35612, offset 0, flags [none],
length: 84) 72.21.34.42 > 69.254.213.43: icmp 64: echo reply seq 0
        0x0000:  4520 0054 8b1c 0000 4001 6a04 4815 222a  E..T....@.j.H."
        0x0010:  45fe d52b 0000 3d30 272a 0000 25ff d744  E..+..=0'..%..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
2 packets captured
2 packets received by filter
0 packets dropped by kernel
hermes root #
```

Examples

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 is indicated by `dir`, and there you can have `src`, `dst`, `src or dst`, and `src and dst`. Here are a few that you should definitely be comfortable with:

- **host** // look for traffic based on IP address (also works with hostname if you're not using **-n**)

  # tcpdump host 1.2.3.4

- **src**, **dst** // find traffic from only a source or destination (eliminates one side of a **host** conversation)

  # tcpdump src 2.3.4.5
  # tcpdump dst 3.4.5.6

- **net** // capture an entire network using [CIDR](#) notation

  # tcpdump net 1.2.3.0/24

- **proto** // works for tcp, udp, and icmp. Note that you don't have to type `proto`

  # tcpdump icmp

- **port** // see only traffic to or from a certain port

  # tcpdump port 3389

- **src, dst port** // filter based on the source or destination port

  # tcpdump src port 1025 # tcpdump dst port 389

- **src/dst, port, protocol** // combine all three

  # tcpdump src port 1025 and tcp

```
# tcpdump udp and src port 53
```

You also have the option to filter by a *range* of ports instead of declaring them individually, and to only see packets that are above or below a certain size.

- **Port Ranges** // see traffic to any port in a range
  tcpdump portrange 21-23

- **Packet Size Filter** // only see packets below or above a certain size (in bytes)
  tcpdump less 32
  tcpdump greater 128

  [You can use the symbols for *less than*, *greater than*, and *less than or equal* / *greater than or equal signs* as well.]

  // filtering for size using symbols
  tcpdump > 32
  tcpdump <= 128

Writing to a File

`tcpdump` allows you to send what you're capturing to a file for later use using the **-w** option, and then to read it back using the **-r** option. This is an excellent way to capture raw traffic and then run it through various tools later.

The traffic captured in this way is stored in `tcpdump` format, which is pretty much universal in the network analysis space. This means it can be read in by all sorts of tools, including [Wireshark](#), [Snort](#), etc.

Capture all Port 80 Traffic to a File

tcpdump -s 1514 port 80 -w capture_file

Then, at some point in the future, you can then read the traffic back in like so:

Read Captured Traffic back into `tcpdump`

tcpdump -r capture_file

Getting Creative

Expressions are nice, but the real magic of `tcpdump` comes from the ability to *combine* them 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 computers at all they'll be pretty familar to you:

1. **AND**
   and or &&

2. **OR**
   or or **||**
3. **EXCEPT**
   not or **!**

More Examples

\# TCP traffic from 10.5.2.3 destined for port 3389

tcpdump -nnvvS src 10.5.2.3 and dst port 3389

\# Traffic originating from the 192.168 network headed for the 10 or 172.16 networks

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 destined for 192.168.0.2 from the 172.16 network

tcpdump -nvvXSs 1514 dst 192.168.0.2 and src net and not icmp

\# Traffic originating from Mars or Pluto that isn't to the SSH port

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.

Grouping

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 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)'

Advanced

You can also filter based on specific portions of a packet, as well as combine multiple

conditions into groups. The former is useful when looking for only SYNs or RSTs, for example, and the latter for even more advanced traffic isolation.

[Hint: An anagram for the TCP flags: **U**nskilled **A**ttackers **P**ester **R**eal **S**ecurity **F**olk]

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]

Keep in mind the reasons these filters work. The filters above 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.

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.

Capture TCP Flags Using the `tcpflags` Option…

tcpdump 'tcp[tcpflags] & & tcp-syn != 0'

Specialized Traffic

Finally, there are a few quick recipes you'll want to remember for catching specific and specialized traffic, such as IPv6 and malformed/likely-malicious packets.

IPv6 traffic

tcpdump ip6

Packets with both the RST and SYN flags set (why?)

tcpdump 'tcp[13] = 6'

Traffic with the 'Evil Bit' Set

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

Conclusion

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.

[If you liked this, check out my other technical primers here.]

Notes

1. I'm currently writing a book on tcpdump for No Starch Press.
2. The leading image is from SecurityWizardry.com.

Recommended for you:

- An ICMP Reference
- Entering Promiscuous Mode on OS X
- Not All SYNs Are Created Equal
- How to Remember Your TCP Flags
- Benign Indian DDoS Attempt
- Building a Professional Firewall with Linux and Iptables

[NOTE: If you liked this page, you should check out my tutorials].

Connect…

:: RSS
:: Twitter
:: Github

Get the weekly summary…

Your email here…

Subscribe

24 Comments   **danielmiessler.com**   1   Login ▾

♥ Recommend 9   ↱ Share   Sort by Best ▾

Join the discussion…

khailey · 2 years ago

nice to see a writeup on tcpdump.

Linked it into my blog post on pinpointing the layer responsible for majority of latency in
NFS using tcpdump to capture, wireshark to translate and perl to parse, agregate and
filter http://www.kylehailey.com/tcp-...

5 ∧ │ ∨ · Reply · Share ›

Russell · 2 years ago

Great write up. Just FYI, it does not look like this is correct.

Traffic originating from Mars or Pluto that isn't to the SSH port

tcpdump -vv src mars and not dst port 22

2 ∧ │ ∨ · Reply · Share ›

Nope · 2 years ago

Nice job! Thanks so much for all that you do!

2 ∧ │ ∨ · Reply · Share ›

uday3110 · 2 years ago

Amazingly concise and informative

1 ∧ │ ∨ · Reply · Share ›

Dan · 2 years ago

Nice tutorials, thanks for writing these up! Definitely bookmarked.

1 ∧ │ ∨ · Reply · Share ›

Vincent Bernat · 2 years ago

The default value for `-s` is 65535 since 4.1.0. However, I keep using `-s 0` to not care of
the version. Also, using more than 1514 is useful even without jumbo frames, for example
when you are using TSO/GSO/GRO/LRO.

I also like to use a command like this:

```
ssh somehost tcpdump -s0 -w - -U some filter | wireshark -k -i -
```

1 ∧ │ ∨ · Reply · Share ›

Mike · 2 years ago

Nice one. I think -s0 is even better if you want to capture all packet data (jumbo frames like mentioned before)

1 ∧ | ∨ • Reply • Share ›

Tomer Ben David · 9 days ago

is tcpdump also suitable for SSL communication? (I want to see it unencrypted is it possible with tcpdump somehow)

∧ | ∨ • Reply • Share ›

POOL-POG · 3 months ago

Thanks for this article. You should consider fixing the "curly quotes" in this article, as they cause problems with copy-n-paste.

∧ | ∨ • Reply • Share ›

Manesti · 3 months ago

Nice, Good Stuff!

∧ | ∨ • Reply • Share ›

ram · 5 months ago

I want to filter udp packets from tcpdump file and getting output removing ntp and dns packets. I tried to filter packets with dst and src ports for 123 and 53. Below is the command.

tcpdump -r filename -w outputfile 'udp and (not src port 53 or src port 123 or dst port 53 or dst port 123)'

But still when I search the output dump file I saw NTP packets. Could some one explain me the wrong doing of my command.

∧ | ∨ • Reply • Share ›

superjamie · 10 months ago

I agree with your statement that a deep understanding of protocols is required to troubleshoot issues, people always come to me saying "How do I read packet captures?" and I reply "Have you got two days to learn about the OSI model, Ethernet, IP, TCP and UDP, plus how NICs, switches, and routers work?". One needs to understand all these concepts before a packet capture with a duplicate IP or TCP Fast Retransmission becomes understandable.

However, I think you go about it the wrong way.

Learning "tcp[13] &2!=0" is fairly specific to tcpdump, and the logical "AND Not Equal To" is opaque to many. Wireshark's "tcp.flags.syn eq 1" is much more intrinsic and approachable. Wireshark also has an arguably superior way of displaying and expanding its protocol dissectors, plus it lets you right click and "Prepare filter". Both of these can enable learning of protocols and filtering of desired information through visual inspection

of intrinsic protocol terms, rather than learning random commands full of symbols and numbers you'll never use anywhere else.

∧ | ∨ • Reply • Share ›

Andres Maye · 10 months ago

Hi, nice tutorial. Do you have something similar, but explaining the "outputs" itself? I mean, the output of tcpdump is sometimes "tricky" to understand in between so many fields, flags, codes, etc.

∧ | ∨ • Reply • Share ›

sisila priyankara · 10 months ago

nice tcpdump tutorial ...

∧ | ∨ • Reply • Share ›

Mallika · 10 months ago

Single quotes are used in order to tell tcpdump to ignore certain special characters — in this case the "( )" brackets.

I'd assume tcpdump needs to know about them - it's the shell that needs to ignore them.

∧ | ∨ • Reply • Share ›

VURSO · a year ago

Nice high quality write up dude, appreciated

∧ | ∨ • Reply • Share ›

HPack · a year ago

There's a typo, Unskilled Attackers Pester Real SecurityFolk is a mnemonic, not an anagram.

∧ | ∨ • Reply • Share ›

Tomaz · a year ago

On the example labeled "Capture TCP flags using the tcpflags option" you write it like so:

tcpdump 'tcp[tcpflags] && tcp-syn != 0'

You're using && instead of & . Since we're dealing with flags on bits, maybe you've made a mistake using the boolean AND instead of the bitwise AND ?

Other than that, awesome and very clear article. O/

∧ | ∨ • Reply • Share ›

hooman · a year ago

thanks man , it is very helpful to me

∧ | ∨ • Reply • Share ›

Clif · a year ago

Thanks for a the tcpdump tutorial/primer. Very helpful and easy to understand.

kousi · 2 years ago

great post

∧ | ∨ · Reply · Share ›

Dissector Writer · 2 years ago

In a discipline so dependent on a true understanding of concepts vs.rote learning, it's important to stay fluent in the underlying mechanics of the TCP/IP suite. A thorough grasp of these protocols allows one to troubleshoot at a level far beyond the average analyst, but mastery of the protocols is only possible through continued exposure to them.

Let's say you have a 1GB capture and you need to know how many SMB2 CREATEs that wanted WDAC access compared with those that did not want WDAC access, how are you going to do this with tcpdump?

∧ | ∨ · Reply · Share ›

hanruna · 2 years ago

Always (years) wanted to find something precisely like this. Somewhere to begin!

∧ | ∨ · Reply · Share ›

Marcin Cabaj · 2 years ago

'-ttt' is a nice option, prints delta time between packets

∧ | ∨ · Reply · Share ›

ALSO ON **DANIELMIESSLER.COM**   WHAT'S THIS?

Serialization Security Bugs Explained

1 comment • 20 days ago

> **FuzzyMind** — A good write up. But Would be great if this can be illustrated with an example.

[Web Scanner Series] Burp vs. Netsparker

3 comments • 2 months ago

> **Loki** — Thank you! Great help!

In the U.S., Gun Violence Might Mostly Mean Suicide

2 comments • 25 days ago

> **Daniel Miessler** — I strongly agree.

Defeating Trump Might Required "A Few Good Men"

2 comments • a month ago

> **okay** — I never reply to political blog posts but I like how you think and I learn from you even when I …

✉ Subscribe   Ⓓ Add Disqus to your site Add Disqus Add   🔒 Privacy

Search this website …   Search

:: RSS
:: Twitter
:: Github

Recommended for you…

- [An ICMP Reference](#)
- [Entering Promiscuous Mode on OS X](#)
- [Not All SYNs Are Created Equal](#)
- [How to Remember Your TCP Flags](#)
- [Benign Indian DDoS Attempt](#)
- [Building a Professional Firewall with Linux and Iptables](#)

Recent Posts

- [Palo Alto Firewalls Can't Update on Leap Year Day](#)
- [My Response to Sam Harris on the Apple Encryption Debate](#)
- [Mandiant M-Trends 2016 Summary](#)
- [Great Leaders Make People Feel Safe](#)
- [T1SP: Episode 29](#)

Get the weekly summary…

Your email here…
Subscribe

Listen to the podcast…

1. [Subscribe via iTunes](#)
2. [Podcast RSS Feed](#)

Categories

- [Business](#) (45)

- [Creativity](#) (147)
- [Culture](#) (244)
- [Happiness](#) (213)
- [Information Security](#) (308)
- [Personal](#) (76)
- [Philosophy](#) (243)
- [Politics](#) (208)
- [Science](#) (119)
- [Take 1 Security Podcast](#) (31)
- [Technology](#) (584)

Archives

Archives Select Month ▼

© Daniel Miessler 1999-2016 | [Stack](#) | [Share](#) | [Syndication](#) | [Privacy](#) | [About the Ads](#) |
[Media](#)