

## Tcpdump - some recipes

Reproduced with permission from Daniel Miessler. 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 ...U.....
```

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 wierdness in the sequence numbers if they're beeing 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 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 usually give it 1514 (to get everything) if I use this option. Here's a short list of the options I use most:

- -n : Don't resolve hostnames.
- -nn : Don't resolve hostnames or port names.
- -X : Show the packet's contents in both hex and ASCII.
- -v, -vv, -vvv : Increase the amount of packet information you get back.
- -c : Only get x number of packets and then stop.
- -S : Print absolute sequence numbers.
- -e : Get the ethernet header as well. 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:

- Basic communication // see the basics without many options

```
tcpdump -nS
```

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

```
tcpdump -nnvvS
```

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

```
tcpdump -nnvvXS
```

- 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 1514 -c2 icmp

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 1514
bytes 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 #

```

Expressions 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 3389

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 familiar to you:

- AND  
and or &&  
- OR  
or or ||  
- EXCEPT

not or ! TCP traffic from 10.5.2.3 destined for port 3389:

```
# tcpdump -nvvS tcp and 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 172.16.0.0/16 and not icmp
```

Traffic originating from Mars or Pluto that isn't to the SSH port: // requires name resolution

```
# tcpdump -vv src mars or pluto 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. 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: // wrong

```
# 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: Unskilled Attackers Pester Real Security Folk

Show me all URG packets:

```
# tcpdump 'tcp[13] & 32 != 0' Show me all ACK packets:
```

```
# tcpdump 'tcp[13] & 16 != 0' Show me all PSH packets:
```

```
# tcpdump 'tcp[13] & 8 != 0' Show me all RST packets:  
# tcpdump 'tcp[13] & 4 != 0' Show me all SYN packets:  
# tcpdump 'tcp[13] & 2 != 0' Show me all FIN packets:  
# tcpdump 'tcp[13] & 1 != 0' Show me all SYN-ACK 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.

Specialized Traffic

- Display all IPv6 Traffic:

```
# tcpdump ip6
```

- Show all traffic with both SYN and RST flags set: (should never happen)

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

- Show all 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.:

Related Articles [A Collection of Tcpdump Recipes](#)

[http://dmiessler.com/study/tcpdump\\_recipes](http://dmiessler.com/study/tcpdump_recipes) [How To Remember Your TCP Flags](#)

<http://dmiessler.com/study/tcpflags> [Not All SYN Packets Are Created Equal](#)

<http://dmiessler.com/study/synpackets>

References [Tcpdump Manual Page](#)

[http://www.tcpdump.org/tcpdump\\_man.html](http://www.tcpdump.org/tcpdump_man.html)