WHAT ARE THOSE COMPUTERS SAYING TO EACH OTHER?

WITH TCPDUMP, YOU CAN FIND OUT!
what’s this?

The man page for tcpdump starts like this:

NAME

tcpdump - dump traffic on a network

SYNOPSIS

tcpdump [-AbdDefhHIJKllNoPpqStuVvxX#] [-B buffer_size]
[[-c count ]
[[-C file_size ] [-G rotate_seconds ] [-F file ]
[-i interface ] [-j tstamp_type ] [-m module ] [-M secret ]
[--number ] [-Q in|out|inout ]
[-r file ] [-V file ] [-s snaplen ] [-T type ] [-w file ]
[-W filecount ]
[-E spi@ipaddr algo:secret.... ]
[-y datalinktype ] [-z postrotate-command ] [-Z user ]
[--time-stamp-precision=tstamp_precision ]
[--immediate-mode ] [ --version ]
[ expression ]
what is tcpdump for?

tcpdump captures network traffic and prints it out for you.

For example! Yesterday DNS lookups on my laptop were slow

```
what's happening?
I know, I'll use tcpdump!
```

```
$ sudo tcpdump -n -i any port 53
```

```
```

This means that there were 3 DNS queries (at 10:52:03, 10:52:08, 10:52:13), but only the 3rd one got a response!

I figured my router was probably the problem, I restarted it, and my internet was fast again!

Let's learn how to debug problems with tcpdump!
Questions you can answer with `tcpdump`

→ what DNS queries is my laptop sending?
   "`tcpdump -i any port 53`"

→ I have a server running on port 1337. Are any packets arriving at that port at ALL???
   "`tcpdump -i any port 1337`"

→ What packets are coming into my server from IP 1.2.3.4?
   "`tcpdump port 1337 and host 1.2.3.4`"

→ Show me all DNS queries that fail
   "`tcpdump udp[17] &0xf == 3`"
   (complicated but it works !)

→ how long are the TCP connections on this box lasting right now?
   "`tcpdump -w packets.pcap`"
   and analyze `packets.pcap` in Wireshark
what tcpdump output means

Every line of tcpdump output represents a packet. The parts I usually pay attention to are:

* source + dest IP address and port
* timestamp
* which TCP flags (good for spotting the beginning of a TCP connection)
* the DNS query, for DNS packets
* that's it!

**UDP packet:**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source IP</th>
<th>dest IP</th>
<th>Port</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:52:03.992138</td>
<td>192.168.1.241.63019</td>
<td>192.168.1.1.53</td>
<td>44000+</td>
<td></td>
</tr>
</tbody>
</table>


**TCP packet:**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source IP</th>
<th>dest IP</th>
<th>Port</th>
<th>Port</th>
</tr>
</thead>
</table>

ack 2291349910, win 319, options [nop,nop,TS val 10967552 ecr 580196754], length 0

Ever seen a "Connection refused" error? Here's what that looks like in tcpdump!

SYN

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source IP</th>
<th>dest IP</th>
<th>Port</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:16:38.944390</td>
<td>localhost.48680</td>
<td>localhost.8999</td>
<td>Flags [S]</td>
<td></td>
</tr>
</tbody>
</table>

12:16:38.944458 | localhost.8999 | localhost.48680 | Flags [R.] |

RST  ACK

We sent a SYN to open the connection but the server replied with a "RST" packet. That gets translated to "connection refused".
BPF filters!

tcpdump uses a small language called BPF to let you filter packets.

When you run `$ sudo tcpdump port 53`, "port 53" is a BPF filter. Here's a quick guide!

→ port 53
checks if the source port OR the dest port is 53. Matches TCP port 53 and UDP port 53.

→ host 192.168.3.2
checks if the source or dest IP is 192.168.3.2

→ host 11.22.33.44 and port 80
you can use 'and', 'or', and 'not'

→ src port 80
→ dest port 80
→ tcp port 80
are what they look like ☺ so are
src host 1.2.3.4
dest host 1.2.3.4

you can do bit math like this on packet contents.
This checks for the DNS response code "NXDOMAIN"!

(I googled to find this and it works! 😊)
Wireshark is an incredibly powerful packet analysis tool!

What protocols do you understand, Wireshark?

HTTP! TCP! DNS! ARP! IP!
MSN! AIM! AOL!
Ethernet! Bluetooth!
A lot, okay?

Things Wireshark has:

★ nice graphical interface!
★ it can connect TCP packets from the same connection!
★ search through your packets easily!

If you want to analyze packets from tcpdump with Wireshark, you can either:

1. save a .pcap file and open it with Wireshark
2. use this incantation to pipe tcpdump output into Wireshark!

ssh some.remote.host tcpdump -pni any -w - -s0 -U port 8888 | wireshark -k -i -
my favourite command line arguments

I use these 3 arguments the most:

- **-i** is for interface
  Which network interface to capture packets on. I often use **-i any**. The default interface tcpdump picks isn’t always what you want.
  
  Example: sudo tcpdump -i lo shows you packets on the local “loopback” interface.

- **-w** is for write
  Instead of printing out packets, write them to a file! This is VERY USEFUL for analyzing the packets later. I use it all the time
  
  Example: sudo tcpdump host 8.8.8.8 -w my_packets.pcap saves packets to/from 8.8.8.8 to a file

- **-c** is for count
  When writing to a file, be careful! You don’t want to accidentally fill up your hard drive. **-c 10000** will only capture 10,000 packets.
  
  Example: sudo tcpdump -c 1000 -w my_packets.pcap dest port 8080
and here are a few more good ones:

This prints out the packet's contents! For example, suppose I have a webserver on port 7777.

```
$ sudo tcpdump -A dest port 7777
```

will show me all the HTTP requests being sent to that server. Only works for HTTP, not HTTPS.

(I like `ngrep` more than `tcpdump -A` for looking at HTTP request bodies though °)

By default, `tcpdump` will translate IP addresses to hostnames. `-n` forces it to just always print out the IP address

Includes Ethernet information! This shows you the MAC address that the packet came from

Example: `sudo tcpdump -e -i any port 443`

makes sure you only get packets that are to or from your computer
network administration tools

Finally, there are a lot more tools than tcpdump! We won't explain them here but here's a list!

**ping**
"are these computers even connected?"

**dig/nslookup**
"does that domain exist?"

**netstat/ss**
"am I using that port?"

**ifconfig**
"what's my IP address?"

**ip**
configures interfaces, routes, and more. Successor to if config.

**arp**
See your ARP table!

**ngrep**
grep for your network

**nftables/iptables**
set up firewalls and NAT!

**trace route/mtr**
what servers are on the way to that server?

**nc**
netcat! Make TCP connections manually!

**sysctl**
configure socket buffer sizes, and more!

**ethtool**
understand your ethernet connections
nmap
in ur network scanning ur ports

telnet
See if a port on another server is open

whois
look up a domain

ssh
can't forget this one!

lsot
what ports are being used?

sysctl
configure socket buffer sizes, and more!

network manager
GUI tool to configure the network on your laptop

ping
ping, but it uses TCP

nethogs/ab/nload
iperf/netperf/iperf
iftop/netsniff-ng
lots of performance/benchmarking tools (they all do different things)

OpenVPN
Set up a VPN!

socat
like netcat, but more featureful

thanks so much for reading!

now that I understand the basics, the man page isn't so bad!
like this?
there are more
zines at:
http://jvns.ca/zines