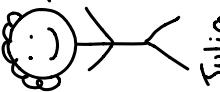


love this?
more zines at
→ wizardzines.com ←

hello!

quick note: 5 of
these tools are Linux-only



Julia

Linux only tool BSD/ Mac equivalent

ip

ifconfig, route

tc

dummynet (?) (BSD)

ss

netstat

iptables

pf (BSD)

ethtool

ifconfig, kind of (?)

23

ethtool

ethtool is for people
who need to manage
physical networks

why no internet?
oops! it would help
server
(if your ethernet
cable was plugged in)

ethtool eth0
name of
network interface

this tells you:

- is it even connected?
- ("link detected")
- speed
- lots more

-- show-offload
-- offload
your network card can
do a lot for you! Like
computing checksums. This
is called "offloading".
This lets you see / change
configured offloads.

iw dev wlan0 link

ethtool is mostly for
Ethernet.

To see the speed (and more)
of a wireless connection,
use iw.

-- identify INTERFACE

blink the light on the
ethernet port. good if you
have multiple ports! and cute!

- S INTERFACE

show statistics like bytes sent.
works for wifi interfaces too.

-S
Change speed/duplex / other
settings of an interface
\$ ethtool -s eth0 speed 100

-i INTERFACE
show firmware info

conntrack

conntrack is used for:

- NAT (in a router !)

- firewalls (eg only allow outbound connections)

- src + dest IP

It's a kernel module called `nf_conntrack`

how to enable conntrack

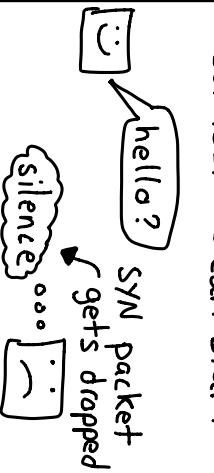
enable:

```
$ sudo modprobe nf_conntrack
```

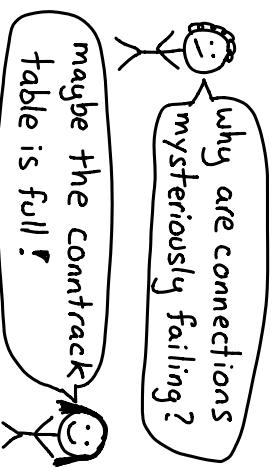
check if it's enabled:

```
$ lsmod | grep conntrack  
change table size with the sysctl  
net.netfilter.nf_conntrack_max
```

if the conntrack table gets full, no new connections can start



moral: be careful about enabling conntrack!



♥ Table of contents ♥

dig	4	tcpdump	10-11	ip	18
ping	5	tshark	12	ss / netstat	19
curl	6	ngrep	13	iptables	20
nmap	7	openssl	14	tc	21
netcat	8	mitmproxy	15	conntrack	22
socat	9	misc tools	16	ethtool	23
ssh	17				

conntrack has a table of every connection

Each entry contains:

- src + dest IP

- src + dest ports

- the connection state (eg TIME_WAIT)

dig

4

dig makes DNS queries!

```
$ dig google.com
answers have 5 parts:
query: google.com
TTL: 22
class: IN (for "internet")
ignore this
record type: A
record value: 172.217.13.110
```

dig TYPE domain.com

this lets you choose which DNS record to query for!
 types to try: NS default
 MX CNAME A
 TXT

dig @ 8.8.8.8 domain

"Google DNS server"
 dig @server lets you pick which DNS server to query! Useful when your system DNS is misbehaving :)

dig +trace domain

traces how the domain gets resolved, starting at the root nameservers if you just updated DNS, dig +trace should show the new record

dig -x 172.217.13.174

makes a reverse DNS query - find which domain resolves to an IP! Same as
 dig ptr 174.13.217.172.in-addr.arpa

21

tc

tc

is for "traffic control"
 packets! stop/slow down/ go the other way!
 great for simulating network problems!

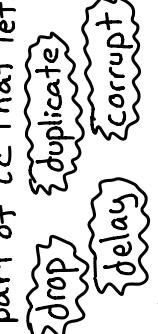


make your internet slow

```
$ sudo tc qdisc add dev
wlp3s0 root netem
delay 500ms <-- delay packets by 500ms
and fast again.
$ sudo tc qdisc del dev
wlp3s0 root netem
```

netem rules

netem ("network emulator") is a part of tc that lets you:
 drop duplicate corrupt delay

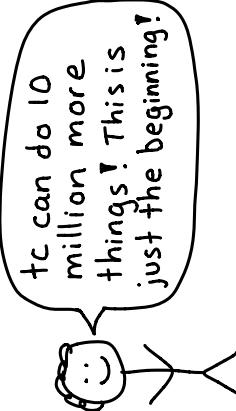


make your brother's internet slow

Have a Linux router? You can configure tc on it to make your brother's internet slower than yours
 google: "tc QoS" for a start

show current tc settings

```
$ tc qdisc show
$ tc class show dev DEV
$ tc filter show dev DEV
```



iptables

iptables-save

This prints out all iptables rules. You can restore them with `iptables-restore`

It's used for firewalls and NAT

iptables lets you create rules to match network packets and accept/drop/modify them

tables have chains
chains have rules

tables: filter, nat, mangle, raw, security
chains: INPUT, FORWARD, PREROUTING, etc

rules: like `-s 10.0.0.0/8 -j DROP`

-j TARGET

Every iptables rule has a target (what to do with matching packets). Options:
 → ACCEPT, DROP, RETURN
 → the name of an iptables chain
 → an extension (man iptables-extensions)
 Popular: DNAT, LOG, MASQUERADE

tables have different chains
 filter: INPUT, OUTPUT, FORWARD, mangle: INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING
 nat : OUTPUT, PREROUTING, POSTROUTING
 It helps to know when packets get processed by a given table/chain (eg locally generated packets go through filter and OUTPUT)

you can match lots of packet attributes
`-s: src ip -p: tcp/udp`
`-d: dst ip -i: network interface`
`-m: lots of things!`
 (bpf rules! cgroups! ICMP type!
 cpu! conntrack state! more!)
 For more, run:
`$ man iptables-extensions`

Ping & traceroute

5

ping checks if you can reach a host and how long the host took to reply

output:
 ... time=253ms ...
 Australia is 17,000 km from me
 at the speed of light it's still far!

ping works by sending an ICMP packet and waiting for a reply

to: health.gov.au
 hello!

I'm here! — health.gov.au

myth: if a host doesn't reply to ping, that means it's down
 Some hosts never respond to ICMP packets. This is why traceroute shows "... sometimes." Sometimes, a host is not listening.

ping {not listening!} host

traceroute tells you the path a packet takes to get to a destination

me) NYC Sacramento Australia
 my ISP

example traceroute

```
$ traceroute health.gov.au
```

```
1: 192.168.1.1 3ms ← router
```

```
2: ...yul.ebox.ca 12 ms ← ISP
```

```
... 8: NYC4.ALTER.NET 24 ms } ← crossings + time spent in routers
```

```
9: SAC1.ALTER.NET 97 ms }
```

```
16: health.gov.au 253ms
```

here the packet crossed the USA!

traceroute health.gov.au
 1: 192.168.1.1 3ms ← router
 2: ...yul.ebox.ca 12 ms ← ISP
 ... 8: NYC4.ALTER.NET 24 ms } ← crossings + time spent in routers
 9: SAC1.ALTER.NET 97 ms }
 16: health.gov.au 253ms

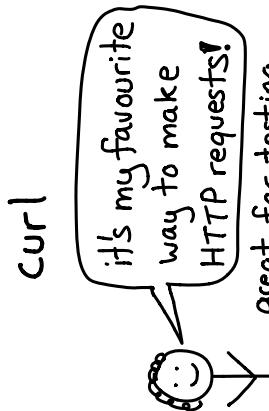
here the packet crossed the USA!

mtr
 like traceroute, but nicer output! try it!

look up how traceroute works (using TTLs!) it's simple + cool!

Curl

6



<code>-H</code>	is for <u>header</u> good for <u>POST</u> requests to JSON APIs: <code>-H "Content-Type: application/json"</code>
<code>-L</code>	allow <u>compressed response</u> : <code>-H "Accept-Encoding: gzip"</code>
<code>-V</code>	show request headers & more
<code>-K</code>	insecure: don't verify SSL certificates
<code>--connect-to ::IP</code>	send request to IP instead. <small>or host name use before changing DNS to a new IP</small>

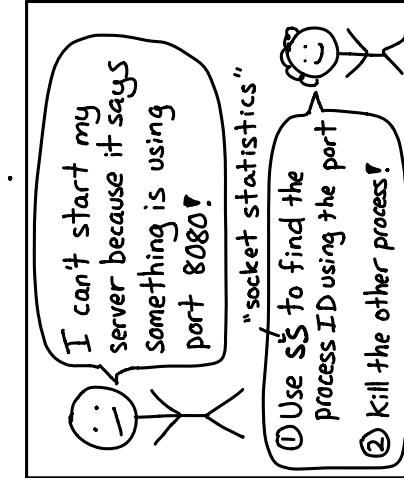
<code>--data</code>	to POST data! <small>--data '{"name": "julia"}'</small>
<code>--data @filename.json</code>	@ reads the data to send from a file

<code>-i</code>	show response headers
<code>-T</code>	show <u>only</u> response headers (makes a HEAD request)
<code>-X POST</code>	send a POST request instead of a GET (-X PUT etc works too)

<code>* copy as curl *</code>	Have something in your browser you want to download from the command line? In Firefox / Chrome / Safari: Developer Tools → Network tab → right click on the request → copy as curl (can have sensitive info in cookies!)
-------------------------------	--

19

<code>* tuna, please! *</code>	\$ ss -tunapl <small>'here the 'a' is bring the 'a' bring doesn't'</small>
	This is my favourite way to use ss! It shows all the running servers



<code>-n</code>	use numeric ports (80 not http)
<code>-D</code>	show PIDs using the socket
<code>{i}</code>	TONS of information

<code>netstat</code>	netstat -tunapl and ss -tunapl do the same thing
<code>default: connections</code>	netstat -tunapl and ss -tunapl do the same thing

<code>-l : listening</code>	netstat is older and more complicated. If you're learning now, I'd recommend ss!
<code>-a : both</code>	

<code>-t : TCP</code>	default: all
<code>-u : UDP</code>	
<code>-X : unix domain Sockets</code>	

ip

ip Linux only

lets you view + change network configuration.

\$ ip OBJECT COMMAND

- addr, link
- add, show,
- neigh, etc
- delete, etc

change your MAC address

good for cafes with time limits 😊

```
$ ip link set wlan0 down
$ ip link set eth0 address
3c:49:f4:d1:00:32
$ ip link set wlan0 up
$ service network-manager
restart ← or whatever you use
```

ip addr list
shows ip addresses of your devices. Look for something like this:

- 2: eth0:
- link/ether 3c:97...
- inet 192.168.1.170/24

ip link
network devices! (like eth0)

ip neigh
view/edit the ARP table

ip xfrm

is for IPsec

ip route list
displays the route table.

default via 192.168.1.1 ↗ "my router"
169.240.0.0/16 dev docker0
...
to see all route tables:
\$ ip route list table all

ip route get IP
what route will packets with \$IP take?

-- color
pretty colourful output!

-- brief

show a summary

nmap

7

aggressive scan

nmap -v -A aggressive scanner. nmap.org

port, server version, even OS

-Pn
skip doing a ping scan and assume every host is up. good if hosts block ping (lots!)

nmap lets you explore a network

which ports are open?
which hosts are up?
security people use it a lot!

fast port scan

```
$ nmap -sS -F 192.168.1.0/24
just sends a SYN packet to check if each port is open.
```

I found out which ports my printer has open!



80 http
443 https
631 printer
9100 jetdirect

-F

scan less ports: just the most common ones

-T4 or **-T5**

scan faster by timing out more quickly

check TLS version

and ciphers

check if your server still supports old TLS versions

```
$ nmap
--script ssl-enum-ciphers
--script ssl-wizardzines.com
list all scripts with:
$ nmap --script-help '*'
```

netcat

8

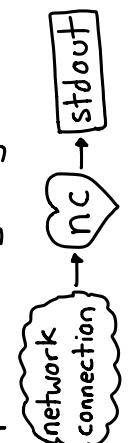
nc

lets you create TCP (or UDP) connections from the command line



nc -l PORT

Start a server! this listens on PORT and prints everything received



nc IP PORT

be a client! opens a TCP connection to IP: PORT (to send UDP use -u)



make HTTP requests by hand

```
$ printf 'GET / HTTP/1.1\r\nHost: example.com\r\n\r\n' | nc example.com 80
```

{ all one line }

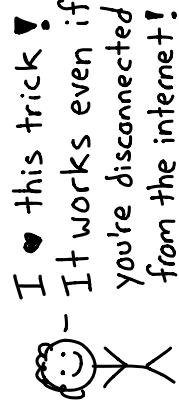
type in any weird HTTP request you want!

want to send a 100 GB file to someone on the same wifi network? easy!

receiver:
\$ nc -l 8080 > file
sender:
\$ cat file.txt | nc YOUR_IP 8080

send files

I ❤️ this trick!
It works even if you're disconnected from the internet!

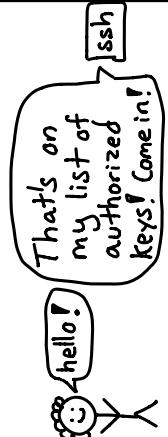


ssh

17

ssh keys ❤️

An ssh key is a secret key that lets you SSH to a machine



ssh user@host -a,
runs this command & exits

ssh-copy-id

This script installs your SSH key on a host (over SSH)
\$ ssh-copy-id user@host
(puts it in .ssh/authorized_keys etc.)



port forwarding *

ssh user@host.com -NfL
33333:localhost:8888
↑
local port
remote port

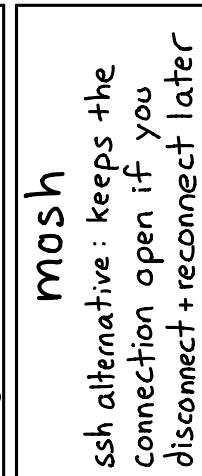
Lets you view a remote server that's not on the internet in your browser.

ssh /config

- ssh user set, per host:
- username to use
- SSH key to use
- an alias!

so you can type \$ ssh ALIAS instead of ssh user@verylongdomain.com

~ •
< Enter> ~. closes the SSH connection. Useful if it's hanging!


mash
ssh alternative: keeps the connection open if you disconnect + reconnect later

just run 1 command

\$ ssh user@host uname -a

remembers your SSH key passphrase so you don't have to keep typing it

miscellaneous networking tools

stunnel make a SSL proxy for an insecure server	rsync sync files over SSH or locally	whois is this domain registered?	zenmap GUI for nmap	sysctl configure Linux kernel's network stack
hping3 make any TCP packet	lsof what ports are being used?	ipcalc easily see what 13.21.2.3/25 means	p0f identify OS of hosts connecting to you	ab / iperf benchmarking tools
wget download files	httppie like curl but friendlier	Python3 -m http.server serve files from a directory	openvpn VPNs	links a browser in your terminal
aria2c a fancier wget	iftop / nettop / nload see what's using bandwidth	nftables new version of iptables	tcpflow capture and assemble TCP streams	telnet can help debug text network protocols
expose a unix domain socket on port 1337	socat supports tcp sockets unix domain sockets pipes SSL sockets files processes UDP sockets ... and MORE!	order doesn't matter socat THING1 THING2 is the same as socat THING2 THING1	-V write all transferred data to stderr useful for debugging!	socat TCP-LISTEN:1337 UNIX-CONNECT:/path
socat TCP-LISTEN:1337 UNIX-CONNECT:/path				

Socat

9

socat lets you proxy basically any 2 things



the basic syntax:
socat THING1 THING2

expose a unix domain socket on port 1337

proxy from local HTTP port to remote server

socat TCP-LISTEN:1337 UNIX-CONNECT:/path

socat TCP-LISTEN:1337 TCP:domain.com:80

tcpdump

10

tcpdump lets you view network packets being sent & received

it's not the easiest to use but it's usually installed ☺

-n

don't try to resolve IP addresses / ports to DNS / port names. makes it run faster.

-w file.pcap

Write packets to a file for later analysis with tcpdump / tshark / Wireshark / another tool

pcap is for "packet capture"

-i wlan0

Which network interface to capture packets on

I often use "-i any" to make sure I'm not missing any packets!

-c 100000

Only capture a limited count of packets

I use it with -w so I don't accidentally fill up my disk!

15

mitmproxy

mitmproxy can proxy connections from your laptop or phone and let you see the contents. It even works with encrypted connections!

???

server

what is my phone saying about me? looks like it's encrypted ...

how you use it

- ① install mitmproxy root CA on your laptop / phone
- ② run mitmweb ↗ web UI on computer
- ③ tell the program / phone to proxy through mitmproxy

other similar tools

(not all are free, though)

- charles proxy
- burp suite
- fiddler

Some apps pin a cert makes mitmproxy not work, look up "trust killer" to get around that

script it in Python

modify requests / responses arbitrarily

how it works

wizardzines.com certificate plz

Yes I am mitmproxy wizardzines.com fake CA you installed

Sounds legit, this CA I trust says that certificate is valid

openssl

inspect a certificate

openssl is a tool for doing *SSL things* aka TLS

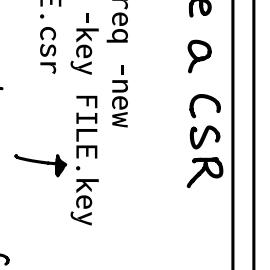
- inspect certificates
- create CSRs
- sign certificates
- It uses the OpenSSL library (or Libressl)





to get a SSL cert for your website, you need to make a file called a "certificate signing request".

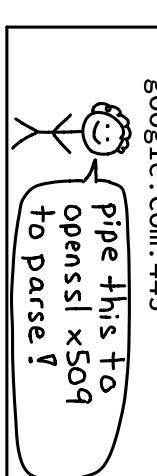
```
$ openssl req -new  
-sha256 -key FILE.key  
-out FILE.csr
```



You probably have certs in /usr/share/ca-certificates

make a CSR

```
$ openssl req -new  
-sha256 -key FILE.key  
-out FILE.csr
```



make one of these with \$ openssl genrsa

md5 / sha1 / sha256 / sha512

Not quite SSL but useful:
\$ openssl md5 FILE computes the md5 sum of FILE. Same for other digests

\$ openssl list -digest-commands shows all supported digests.

BPF cheat sheet

||

host

Filter based on the source or destination IP address

src host google.com ↗
use domain or IP
dst host 192.168.1.1
host 127.0.0.1

same as "src or dst host"

port

src port 53
port 80

again, same as "src or dst port"

less / greater

Packet length!
less 80
greater 200

tcp / udp / icmp

IP4→IP / IPv6

only show packets using that protocol

host 127.0.0.1 and port 80
udp and port 53
(port 53 or port 99) and
not host 127.0.0.1

and / or / not

host 127.0.0.1 and port 80

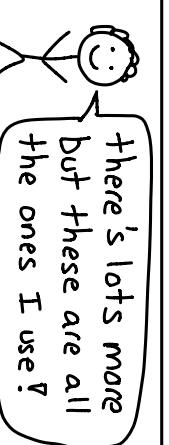
udp and port 53

(port 53 or port 99) and
not host 127.0.0.1

PROTOCOL [INDEX]

filter based on a specific byte in a packet

IP packets with options:
ip[0] & 0xF == 5
DNS SERVFAIL responses:
udp[11] & 0xF > 0
Syn packets:
tcp[tcpflags] == tcp-syn



tshark

12

Wireshark is an amazing graphical packet analysis tool
tshark is the command line version of Wireshark
it can do 100x more things than tcpreplay

-Y
filter which packets are captured
tshark -Y
'http.request.method == "GET'"
↑ uses Wireshark's SUPER POWERFUL filter language

-T FORMAT

Output format. My favourites:
★ json } you can specify fields : csv/tsv } which fields you want with -e
★ text : default summary

-e
Which fields to output. Ex:
\$ tshark -T fields
-e http.request.method
-e http.request.uri
-e ip.dst ↗ supports WAY more protocols than HTTP
GET /foo 92.183.216.34
POST /bar 10.23.38.132

-d
is for "decode as"
tells tshark what protocol to interpret a part as
Example: 8888 is often HTTP!
\$ tshark
-d tcp.port==8888,http

-r file.pcap
analyze packets from a file instead of the network
-w ↗ same as tcpdump's
Write captured packets to a file. If -w file.pcap has permission issues, try:
tshark -w - > file.pcap

ngrep

13

like grep for your network!

\$ sudo ngrep GET
will find every plaintext HTTP GET request

ngrep syntax
what to search
[options] ↓
[regular expression]
[BPF filter] ↓
same format as tcpdump uses!

-d
is for device
which network interface to use. same as tcpdump's
i (try '-d any'?)

-l byline
prints line breaks as line breaks, not "\n".
Nice when looking at HTTP requests

I started using ngrep when I was intimidated by tcpdump and I found it easier ◉



-I file.pcap
-O file.pcap
read/write packets from/to a pcap file