iptables Linux packet filter

START HERE

`sudo iptables -L -n -v --line-numbers`
Show rules with counters

`sudo iptables -S`
Rules as commands

`sudo iptables -A INPUT -p tcp --dport <port> -j ACCEPT`
Allow a port

`sudo iptables -I INPUT 1 -s <ip> -j DROP`
Block a host first

`sudo iptables -D INPUT <num>`
Delete by number

`sudo iptables -t nat -L -n -v`
Show NAT rules

`sudo iptables-save > <file>`
Back up rules

`sudo iptables-restore < <file>`
Load saved rules

PREROUTING raw, mangle, nat (DNAT) → routing → **INPUT / FORWARD** local host or routed → local out → **OUTPUT** locally made packets → routing → **POSTROUTING** mangle, nat (SNAT)

Packet path (simplified): table order differs per chain

CONCEPTS

Rules are checked top to bottom; the first match with a final target (ACCEPT, DROP, REJECT) decides. No match falls to the chain policy.

Changes apply instantly but vanish at reboot unless you save them (e.g. iptables-persistent on Debian). Dumped rules are not loaded on boot by themselves.

A wrong DROP policy over SSH locks you out. Add the ACCEPT rule for your port first, and keep a second session open.

Modern distributions favour nftables; iptables-nft is a compatibility front end over it. See the nftables sheet.

TABLES AND CHAINS

`iptables -t <table> ...`
Pick a table (default filter)

TABLES AND CHAINS (cont.)

Tables and built-in chains

Table	Chains	Used for
`filter`	`INPUT` `FORWARD` `OUTPUT`	Allow or block (default)
`nat`	`PREROUTING` `INPUT` `OUTPUT` `POSTROUTING`	Address translation
`mangle`	`all five`	Alter packets, marks
`raw`	`PREROUTING` `OUTPUT`	Skip connection tracking
`security`	`INPUT` `FORWARD` `OUTPUT`	SELinux-style marks

`sudo iptables -N <chain>`
Create a user chain

`sudo iptables -X <chain>`
Delete an empty user chain
The chain must have no rules and no references to it.

`sudo iptables -P INPUT DROP`
Set default policy

`sudo iptables -A INPUT -j <chain>`
Jump into a user chain

LIST AND INSPECT

`sudo iptables -L`
List rules of the filter table

LIST AND INSPECT (cont.)

`sudo iptables -L INPUT -n -v --line-numbers`
Chain with numbers and counters

`sudo iptables -S <chain>`
Print a chain as commands

`sudo iptables -Z`
Zero all counters

`sudo iptables -C INPUT -p tcp --dport <port> -j ACCEPT`
Test if a rule exists
Exit status 0 means the rule exists; nothing is changed.

`iptables -V`
Show version

ADD AND REMOVE RULES

`sudo iptables -A <chain> <rule>`
Append a rule

`sudo iptables -I <chain> [<num>] <rule>`
Insert (default: top)

`sudo iptables -D <chain> <num>`
Delete by position

`sudo iptables -D <chain> <rule>`
Delete by matching rule

ADD AND REMOVE RULES (cont.)

`sudo iptables -R <chain> <num> <rule>`
Replace a rule

`sudo iptables -F [<chain>]`
Flush rules
Flushing does not reset the policy; with policy DROP this can cut you off.

MATCHES

`-p tcp / udp / icmp`
Match protocol

`-s <ip>[/<mask>]`
Match source address

`-d <ip>[/<mask>]`
Match destination address

`-i <iface>`
Match incoming interface

`-o <iface>`
Match outgoing interface

`! -s <ip>`
Negate a match

`-p tcp --dport <port>`
Match destination port

`-p tcp --sport <port>`
Match source port

`-p icmp --icmp-type echo-request`
Match ICMP type

MATCHES (cont.)

`-m multiport --dports <p1>,<p2>`
Several ports in one rule
At most 15 ports.

`-m conntrack --ctstate ESTABLISHED,RELATED`
Match connection state

`-m comment --comment "<text>"`
Label a rule

`-m limit --limit 5/minute`
Rate-limit matches

TARGETS

`-j ACCEPT`
Let the packet through

`-j DROP`
Discard silently

`-j REJECT --reject-with tcp-reset`
Refuse with a reply
Default reply is icmp-port-unreachable.

`-j RETURN`
Back to the calling chain

`-j LOG --log-prefix "<text>: "`
Log to the kernel log
Non-final: processing continues with the next rule.

NAT AND FORWARDING

`sudo iptables -t nat -A POSTROUTING -o <iface> -j MASQUERADE`
Share one dynamic address

`sudo iptables -t nat -A POSTROUTING -s <net> -j SNAT --to-source <ip>`
Rewrite source address

`sudo iptables -t nat -A PREROUTING -p tcp --dport <port> -j DNAT --to-destination <ip>:<port>`
Forward a port inward
Also needs a FORWARD ACCEPT rule and IP forwarding enabled.

`sudo sysctl -w net.ipv4.ip_forward=1`
Enable IPv4 routing

`sudo iptables -A FORWARD -i <in> -o <out> -j ACCEPT`
Allow routed traffic

SAVE AND RESTORE

`sudo iptables-save`
Dump all tables

`sudo iptables-restore -t < <file>`
Only parse, do not apply

`sudo ip6tables ...`
Same commands for IPv6
IPv4 and IPv6 rulesets are separate.