nftables Linux packet filtering

CONCEPTS

Nothing exists by default: you create your own tables and chains. A table only filters if it has a base chain attached to a hook.

Table family: ip ip6 inet arp bridge netdev
- **Chain** base (hook + priority) or regular
  - **Rule** matches then statements

prerouting dnat, early filter
↓ route
input / forward to host / through
↓ route
output from host
↓ route
postrouting snat, masquerade

Use the inet family to write one ruleset for IPv4 and IPv6 together.

Quote braces and semicolons in the shell ('{ ... }') or escape them with a backslash. In files they need no quoting.

INSPECT

`nft list ruleset`
Print all tables

`nft list tables`
List table names

`nft list table <family> <table>`
One table

`nft list chain <family> <table> <chain>`
One chain

`nft -a list ruleset`
Show rule handles

`nft -n list ruleset`
Numeric ports and addresses

TABLES

`nft add table inet <table>`
Create a table

`nft delete table inet <table>`
Delete table with its content

`nft flush table inet <table>`
Remove all its rules

CHAINS

`nft add chain inet <table> input '{ type filter hook input priority 0; policy drop; }'`
Base chain on a hook

`nft add chain inet <table> <chain>`
Regular chain (jump target)

RULES

`nft add rule inet <table> <chain> <match> <statement>`
Append a rule

`nft insert rule inet <table> <chain> <match> <statement>`
Insert at top of chain

`nft delete rule inet <table> <chain> handle <n>`
Delete a rule by handle

`nft add rule inet <table> input tcp dport 22 accept`
Allow SSH

`nft add rule inet <table> input tcp dport { 80, 443 } accept`
Allow several ports

`nft add rule inet <table> input ip saddr 10.0.0.0/8 drop`
Block a network

`nft add rule inet <table> input iifname lo accept`
Allow loopback

`nft add rule inet <table> input ct state established,related accept`
Allow replies

MATCHES

`ip saddr / ip daddr <addr>`
IPv4 source / destination

`ip6 saddr / ip6 daddr <addr>`
IPv6 source / destination

`tcp dport / udp dport <port>`
Destination port

`iifname "eth0" / oifname "eth0"`
Interface by name

`ct state new`
Connection tracking state

`<field> { a, b, c }`
Match any of several values

STATEMENTS

`accept / drop`
Final verdict

`counter`
Count packets and bytes

`log prefix "<text> "`
Log to the kernel log

SETS AND MAPS

`nft add set inet <table> <set> '{ type ipv4_addr; }'`
Create a named set

`nft add element inet <table> <set> { 1.2.3.4, 5.6.7.8 }`
Add elements

`nft add rule inet <table> input ip saddr @<set> drop`
Use a set in a rule

NAT

`nft add table ip nat`
Table for NAT

`nft add chain ip nat postrouting '{ type nat hook postrouting priority 100; }'`
Source NAT chain

`nft add chain ip nat prerouting '{ type nat hook prerouting priority -100; }'`
Destination NAT chain

`nft add rule ip nat postrouting oifname "eth0" masquerade`
Share an uplink

`nft add rule ip nat prerouting tcp dport 8080 dnat to <ip>:80`
Port forward to a host

SAVE AND RESTORE

`nft list ruleset > <file>`
Save the ruleset

`nft -f <file>`
Load a rule file

`nft -c -f <file>`
Check a file without applying

`nft flush ruleset`
Remove everything

`systemctl enable --now nftables`
Load /etc/nftables.conf at boot