



nftables tutorial

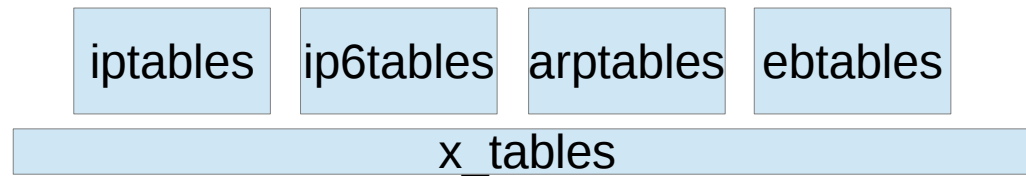
Pablo Neira Ayuso
<pablo@netfilter.org>

Userday Netfilter - June 2016
Netherlands



Why this?

- Abuse of copy and paste from iptables



- In the late 90s people were happy with shell scripts
- Avoid linear ruleset representations: Use concatenations and maps.
- Better incremental updates.
- Simplify dual stack IPv4/IPv6 administration and layer 2.
- No more dash dash spellings.

Bye bye iptables

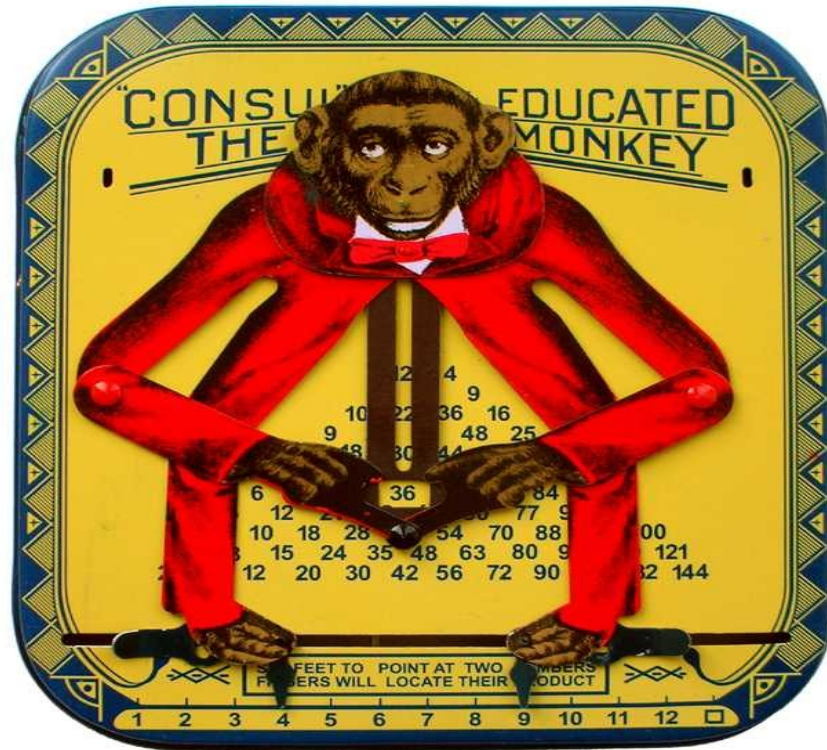


Let's look at internals...

- Simple like...

"Consul" The Educated Monkey.
"Computing Device" by
William H. Robertson (1823-1898)

- Specific purpose VM
 - 22 instructions
 - 32/128 addressable regs
 - Very simple bytecode verifications

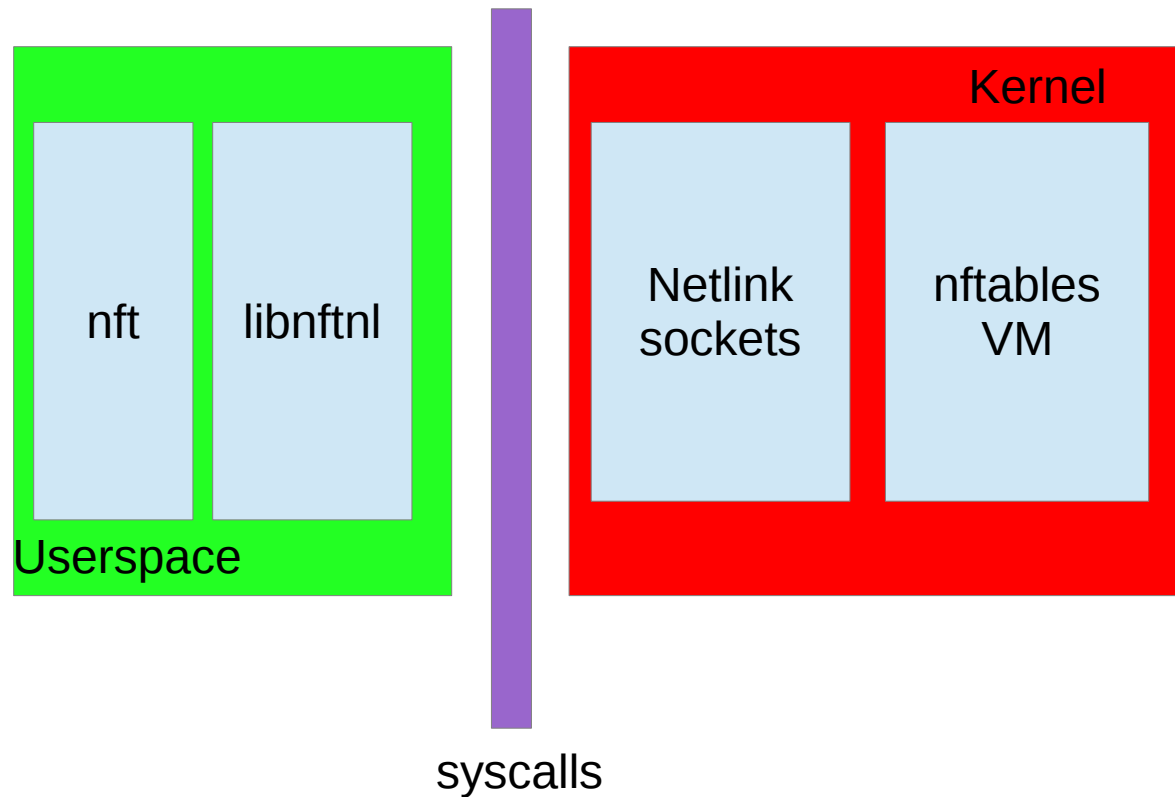


- Extensible like...

- Netlink socket interface



Let's look at internals... (2)



nftables internals

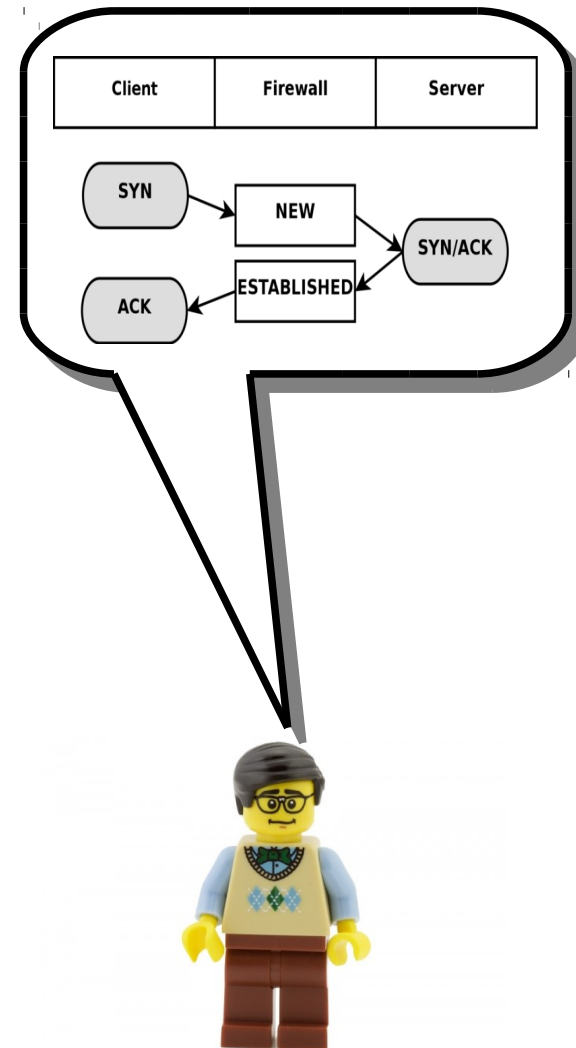
- `nft --debug=netlink add rule ip foo bar ct state new \
ip saddr 192.168.0.0-192.168.0.13 tcp dport 22 accept`
ip foo bar
[ct load state => reg 1]
[bitwise reg 1 = (reg=1 & 0x00000008) ^ 0x00000000]
[cmp neq reg 1 0x00000000]
[payload load 4b @ network header + 12 => reg 1]
[cmp gte reg 1 0x0000a8c0]
[cmp lte reg 1 0x0d00a8c0]
[payload load 1b @ network header + 9 => reg 1]
[cmp eq reg 1 0x00000006]
[payload load 2b @ transport header + 2 => reg 1]
[cmp eq reg 1 0x00001600]
[immediate reg 0 accept]

First off:
specify family,
table and
chain



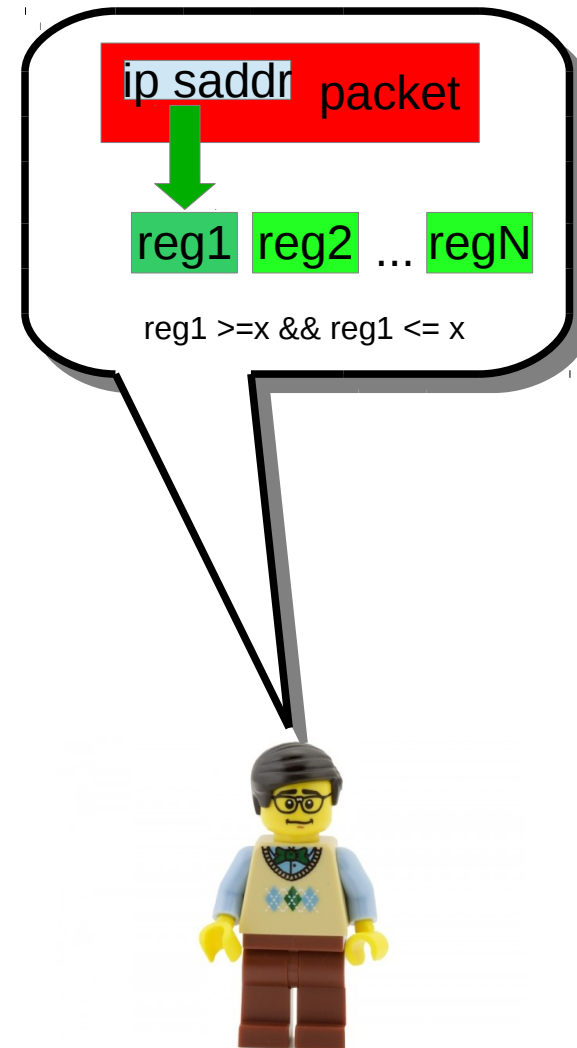
nftables internals

- `nft --debug=netlink add rule ip foo bar ct state new \
ip saddr 192.168.0.0-192.168.0.13 tcp dport 22 accept`
`ip foo bar`
`[ct load state => reg 1]`
`[bitwise reg 1 = (reg=1 & 0x00000008) ^ 0x00000000]`
`[cmp neq reg 1 0x00000000]`
`[payload load 4b @ network header + 12 => reg 1]`
`[cmp gte reg 1 0x0000a8c0]`
`[cmp lte reg 1 0x0d00a8c0]`
`[payload load 1b @ network header + 9 => reg 1]`
`[cmp eq reg 1 0x00000006]`
`[payload load 2b @ transport header + 2 => reg 1]`
`[cmp eq reg 1 0x00001600]`
`[immediate reg 0 accept]`



nftables internals

- `nft --debug=netlink add rule ip foo bar ct state new \`
`ip saddr 192.168.0.0-192.168.0.13 tcp dport 22 accept`
`ip foo bar`
`[ct load state => reg 1]`
`[bitwise reg 1 = (reg=1 & 0x00000008) ^ 0x00000000]`
`[cmp neq reg 1 0x00000000]`
`[payload load 4b @ network header + 12 => reg 1]`
`[cmp gte reg 1 0x0000a8c0]`
`[cmp lte reg 1 0x0d00a8c0]`
`[payload load 1b @ network header + 9 => reg 1]`
`[cmp eq reg 1 0x00000006]`
`[payload load 2b @ transport header + 2 => reg 1]`
`[cmp eq reg 1 0x00001600]`
`[immediate reg 0 accept]`



nftables internals

- nft --debug=netlink add rule ip foo bar ct state new \
ip saddr 192.168.0.0-192.168.0.13 tcp dport 22 accept
ip foo bar
[ct load state => reg 1]
[bitwise reg 1 = (reg=1 & 0x00000008) ^ 0x00000000]
[cmp neq reg 1 0x00000000]
[payload load 4b @ network header + 12 => reg 1]
[cmp gte reg 1 0x0000a8c0]
[cmp lte reg 1 0x0d00a8c0]
[payload load 1b @ network header + 9 => reg 1]
[cmp eq reg 1 0x00000006]
[payload load 2b @ transport header + 2 => reg 1]
[cmp eq reg 1 0x00001600]
[immediate reg 0 accept]

Automatic dependency
generation (in yellow):
ip protocol tcp
and match for
destination port (in blue)



nftables internals

- `nft --debug=netlink add rule ip foo bar ct state new \
ip saddr 192.168.0.0-192.168.0.13 tcp dport 22 accept
ip foo bar
[ct load state => reg 1]
[bitwise reg 1 = (reg=1 & 0x00000008) ^ 0x00000000]
[cmp neq reg 1 0x00000000]
[payload load 4b @ network header + 12 => reg 1]
[cmp gte reg 1 0x0000a8c0]
[cmp lte reg 1 0x0d00a8c0]
[payload load 1b @ network header + 9 => reg 1]
[cmp eq reg 1 0x00000006]
[payload load 2b @ transport header + 2 => reg 1]
[cmp eq reg 1 0x00001600]
[immediate reg 0 accept]`

Ok, let's accept
this packet!



Tables, chains and rules

- nft add table ip foo
- nft add chain ip foo bar { \
 type filter hook input priority 0; policy drop; \
}
- nft add rule ip foo bar \
 ct state established,related accept
nft add rule ip foo bar \
 ct state new tcp dport 22 accept

Tables are empty
by default and they have
no special semantics



Table ip foo

Tables, chains and rules

- nft add table ip foo
- nft add chain ip foo bar { \
type filter hook input priority 0; policy drop; \
}
- nft add rule ip foo bar \
ct state established,related accept
nft add rule ip foo bar \
ct state new tcp dport 22 accept

Base chains see traffic from the specific hook and priority



Table ip foo

Chain bar

Tables, chains and rules

- `nft add table ip foo`
- `nft add chain ip foo bar { \`
 `type filter hook input priority 0; policy drop; \`
 `}`
- `nft add rule ip foo bar \`
 `ct state established,related accept`
`nft add rule ip foo bar \`
 `ct state new tcp dport 22 accept`

You can append new rules and insert them



Table ip foo

Chain bar

Rule

Rule



Expressions

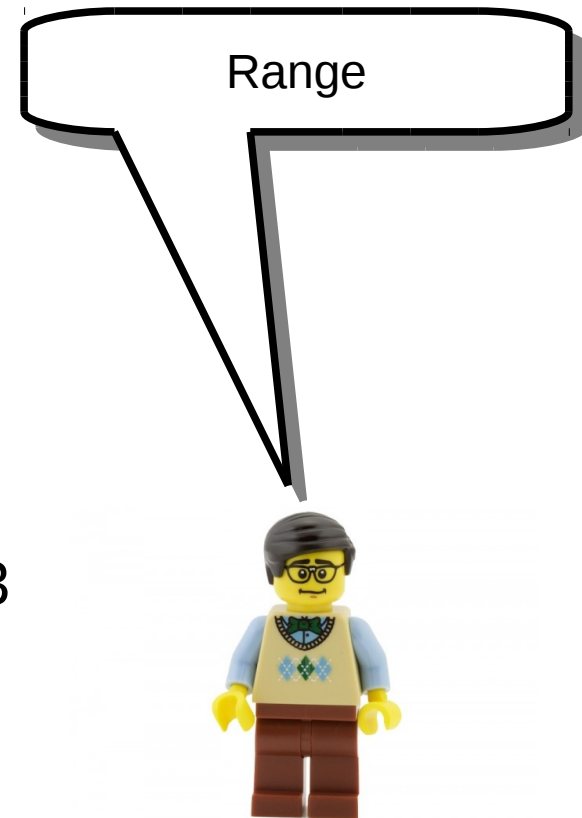
- nft add rule ip foo bar tcp dport != 80
- nft add rule ip foo bar tcp dport 1-1024
- nft add rule ip foo bar meta skuid 1000-1100
- nft add rule ip foo bar ip daddr 192.168.10.0/24
- nft add rule ip foo bar meta mark 0xffffffff/24
- nft add rule ip foo bar ct state new,established
- nft add rule ip foo bar ct mark and 0xffff == 0x123
- nft add rule ip foo bar ct mark set 10
- nft add rule ip foo bar ct mark set meta mark

Comparison: eq, neq,
gt, gte, lt, lte



Expressions

- nft add rule ip foo bar tcp dport != 80
- nft add rule ip foo bar tcp dport 1-1024
- nft add rule ip foo bar meta skuid 1000-1100
- nft add rule ip foo bar ip daddr 192.168.10.0/24
- nft add rule ip foo bar meta mark 0xffffffff/24
- nft add rule ip foo bar ct state new,established
- nft add rule ip foo bar ct mark and 0xffff == 0x123
- nft add rule ip foo bar ct mark set 10
- nft add rule ip foo bar ct mark set meta mark



Expressions

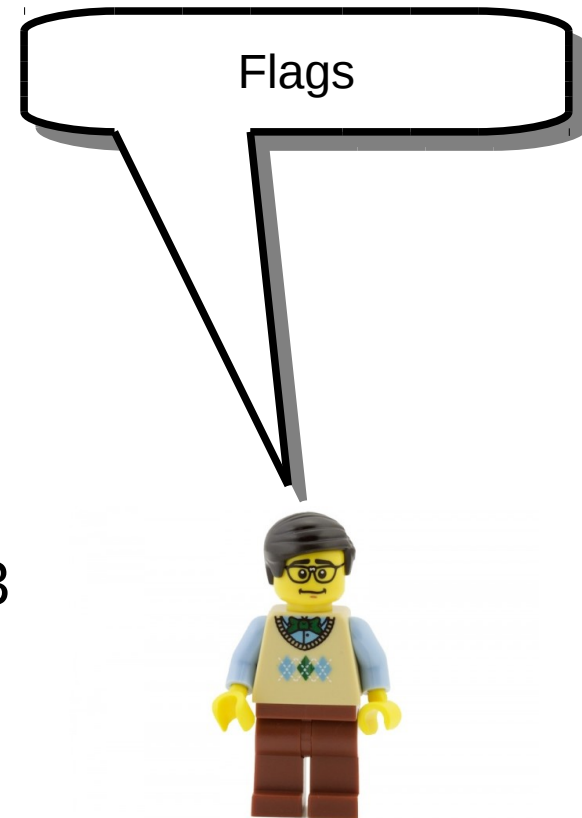
- nft add rule ip foo bar tcp dport != 80
- nft add rule ip foo bar tcp dport 1-1024
- nft add rule ip foo bar meta skuid 1000-1100
- nft add rule ip foo bar ip daddr 192.168.10.0/24
- nft add rule ip foo bar meta mark 0xffffffff/24
- nft add rule ip foo bar ct state new,established
- nft add rule ip foo bar ct mark and 0xffff == 0x123
- nft add rule ip foo bar ct mark set 10
- nft add rule ip foo bar ct mark set meta mark

Prefixes



Expressions

- nft add rule ip foo bar tcp dport != 80
- nft add rule ip foo bar tcp dport 1-1024
- nft add rule ip foo bar meta skuid 1000-1100
- nft add rule ip foo bar ip daddr 192.168.10.0/24
- nft add rule ip foo bar meta mark 0xffffffff/24
- nft add rule ip foo bar ct state new,established
- nft add rule ip foo bar ct mark and 0xffff == 0x123
- nft add rule ip foo bar ct mark set 10
- nft add rule ip foo bar ct mark set meta mark



Expressions

- nft add rule ip foo bar tcp dport != 80
- nft add rule ip foo bar tcp dport 1-1024
- nft add rule ip foo bar meta skuid 1000-1100
- nft add rule ip foo bar ip daddr 192.168.10.0/24
- nft add rule ip foo bar meta mark 0xffff00/24
- nft add rule ip foo bar ct state new,established
- nft add rule ip foo bar ct mark and 0xffff == 0x123
- nft add rule ip foo bar ct mark set 10
- nft add rule ip foo bar ct mark set meta mark

Bitwise + comparison



Expressions

- nft add rule ip foo bar tcp dport != 80
- nft add rule ip foo bar tcp dport 1-1024
- nft add rule ip foo bar meta skuid 1000-1100
- nft add rule ip foo bar ip daddr 192.168.10.0/24
- nft add rule ip foo bar meta mark 0xffffffff/24
- nft add rule ip foo bar ct state new,established
- nft add rule ip foo bar ct mark and 0xffff == 0x123
- nft add rule ip foo bar ct mark set 10
- nft add rule ip foo bar ct mark set meta mark

Set value



Rules

- Counters are optional (unlike iptables)
 - `nft add rule ip foo bar counter`
- Several actions in one rule
 - `nft add rule ip foo bar ct state invalid \`
`log prefix "invalid: " drop`
- Interactive mode (no autocompletion yet)
 - `nft -i`
`nft> add table foo`

I always wanted to
log and drop with
one single rule, heh



Sets and maps

- nft add rule ip foo bar tcp dport { 22, 80, 443 } counter
- nft add set ip foo whitelist { type ipv4_addr \; }
nft add rule ip foo bar ip daddr @whitelist counter accept
nft add element ip foo whitelist { \
 192.168.0.1, \
 192.168.0.10 \
}
- nft add table ip nat
nft add chain ip nat post { \
 type nat hook postrouting priority 0\; }
nft add rule ip nat post snat ip saddr map { \
 1.1.1.0/24 : 192.168.3.11 , \
 2.2.2.0/24 : 192.168.3.12 \
}

The use of brackets from rules result in an implicit set definition



Sets and maps

- nft add rule ip foo bar tcp dport { 22, 80, 443 } counter
- nft add set ip foo whitelist { type ipv4 addr \; }
nft add rule ip foo bar ip daddr @whitelist counter accept
nft add element ip foo whitelist { \
 192.168.0.1, \
 192.168.0.10 \
}
- nft add table ip nat
nft add chain ip nat post { \
 type nat hook postrouting priority 0\; }
nft add rule ip nat post snat ip saddr map { \
 1.1.1.0/24 : 192.168.3.11 , \
 2.2.2.0/24 : 192.168.3.12 \
}

Set declarations require a name and datatype for elements



Sets and maps

- nft add rule ip foo bar tcp dport { 22, 80, 443 } counter
- nft add set ip foo whitelist { type ipv4_addr \; }
nft add rule ip foo bar ip daddr @whitelist counter accept
nft add element ip foo whitelist { \
 192.168.0.1, \
 192.168.0.10 \
}
- nft add table ip nat
nft add chain ip nat post { \
 type nat hook postrouting priority 0\; }
nft add rule ip nat post snat ip saddr map { \
 1.1.1.0/24 : 192.168.3.11 , \
 2.2.2.0/24 : 192.168.3.12 \
}

Refer to an existing set through @



Sets and maps

- nft add rule ip foo bar tcp dport { 22, 80, 443 } counter
- nft add set ip foo whitelist { type ipv4_addr \; }
nft add rule ip foo bar ip daddr @whitelist counter accept
nft add element ip foo whitelist { \
 192.168.0.1, \
 192.168.0.10 \
}
- nft add table ip nat
nft add chain ip nat post { \
 type nat hook postrouting priority 0\; }
nft add rule ip nat post snat ip saddr map { \
 1.1.1.0/24 : 192.168.3.11 , \
 2.2.2.0/24 : 192.168.3.12 \
}

This map allows you to source NAT depending on your source IP address



Set timeouts

- nft add set ip foo whitelist { \
 type ipv4_addr; \
 timeout 1h; \
}
- nft add element ip foo whitelist { \
 192.168.2.123,
 192.168.2.124,
}
- nft add set ip foo whitelist { \
 type ipv4_addr; flags timeout; \
}
- nft add element ip foo whitelist { 192.168.2.123 timeout 10s }

Build your own
whitelists..
Specify global timeouts
for elements or in a
more fine grain fashion



Dictionaries

- nft add chain ip foo tcp-chain
nft add chain ip foo udp-chain
nft add chain ip foo icmp-chain

- nft add rule ip foo bar ip protocol vmap { \
 tcp : jump tcp-chain, \
 udp : jump udp-chain, \
 icmp : jump icmp-chain
}

Jump to non-base chain
based on the layer 4
protocol type



Concatenations

- nft add rule netdev foo bar \
 ether saddr . ip saddr . tcp dport { \
 c0:fe:00:c0:fe:00 . 192.168.1.123 . 80, \
 be:ef:00:be:ef:00 . 192.168.1.120 . 22} \
 counter accept
- nft add rule netdev foo bar ip saddr . tcp dport vmap { \
 192.168.1.123 . 22 : jump whitelist, \
 192.168.1.123 . 80 : jump whitelist, \
}
- nft add set netdev foo bar { \
 type ether_addr . ipv4_addr \;
- nft add element netdev foo bar { \
 00:ca:fe:00:be:ef . 192.168.1.123, \
 00:ab:cd:ef:00:12 . 192.168.1.124 \
}

Concatenate selectors
for fast matching using
dot separated keys
and values

... use this from sets
and maps



Flow tables

- nft add rule ip foo bar ct state new tcp dport 22 \
flow table ssh-spammer { \
 ip saddr limit rate over 3/second
 } log prefix \"New SSH connection: \" drop
- nft list flow table ssh-spammer

... in blue the selector,
in green the action,
and in red the flow
table name.



More actions

```
nft add rule foo bar reject with icmp type host-unreachable
```

```
nft add rule netdev foo ingress \  
  limit rate 10 mbytes/second accept
```

```
nft add rule netdev foo prerouting queue num 3
```

```
nft add rule netdev foo ingress \  
  ether daddr ab:cd:de:ff:00:01 fwd to vethXYZ
```

```
nft add rule netdev foo ingress ip daddr 1.2.3.4 dup to dummy0
```

```
nft add rule nat postrouting snat 1.2.3.4
```

```
nft add rule nat postrouting masquerade
```

```
nft add rule foo prerouting tcp dport 80 tcp dport set 8080
```

Comments

- nft add rule ip foo bar \
ip daddr 8.8.8.8 counter accept\
comment "google dns"
- nft add set ip foo dns-whitelist {\
type ipv4_addr\
}
- nft add element ip foo dns-whitelist { \
8.8.8.8 comment "google dns", \
192.203.230.10 comment "nasa dns",
}

Don't forget
why things are there...



Scripting

Include other ruleset files
and
define variables

```
#!/usr/sbin/nft
```

```
include "another-ruleset.nft"
```

Don't use shell scripts,
use our native scripts!



```
#
```

```
# Allowed NTP servers
```

```
#
```

```
define ntp_servers = { 84.77.40.132, 176.31.53.99, 81.19.96.148,  
138.100.62.8 }
```

```
add rule netdev foo bar ip saddr $ntp_servers udp dport 123 counter
```

Restoring ruleset

- `echo "flush ruleset" > ruleset.nft`
- `nft list ruleset >> ruleset.nft`
- `nft -f ruleset.nft`
- `nft export ruleset json > ruleset.json`

Monitoring update

- `nft monitor`
- `nft monitor new rules`

Tracing

- `nft add rule foo prerouting meta trace 1`
- `nft monitor trace`

Save and restore your ruleset



Monitor ruleset updates

Learn more and help us

- Grab the code
 - Kernel: <http://www.kernel.org>
 - Library: <git://git.netfilter.org/libnftnl>
 - User-space: <git://git.netfilter.org/nftables>
- Documentation
 - <http://wiki.nftables.org>
 - `man nft`
- Report bugs:
 - <https://bugzilla.netfilter.org>
- Follows us @nftables



nftables tutorial

Pablo Neira Ayuso
<pablo@netfilter.org>

Userday Netfilter - June 2016
Netherlands

