TRAINING SESSION NAME

# Firewall

**Table of content**

# A. Definition

## 1. Concept

- Nftables is a netfilter project that aims to replace the existing {ip,ip6,arp,eb} tables framework.
- Main differences with iptables:

+ New syntax: *iptables* command line tool uses a getopt_long()-based parser where keys are always preceded by double minus, eg. --*key* or one single minus, eg. -*p tcp*. In contrast, nftables uses a compact syntax inspired by *tcpdump*.
+ Tables and chains are fully configurable**:** *iptables* has multiple pre-defined tables and base chains, all of which are registered even if you only need one of them. With nftables there are no pre-defined tables or chains.
+ Multiple actions
+ Combine rules

## 2. Configuration

### 2.1 Ruleset

**Definition**

-The ruleset keyword is used to identify the whole set of tables, chains, etc. currently in place in kernel.

### Command

```
{list | flush} ruleset [family]
```

## 2.2 Table

### Definition

- Tables are containers for chains, sets and stateful objects.
- They are identified by their address family and their name.
- Address families determine the type of packets which are processed.

| Address families | Description |
|---|---|
| ip | IPv4 address family. |
| ip6 | IPv6 address family. |
| inet | Internet (IPv4/IPv6) address family. |
| arp | ARP address family, handling IPv4 ARP packets. |
| bridge | Bridge address family, handling packets which traverse a bridge device. |
| netdev | Netdev address family, handling packets on ingress and egress. |

### Command

```
{add | create} table [family] table [ {comment comment ;} { flags 'flags ; }]
{delete | destroy | list | flush} table [family] table
list tables [family]
delete table [family] handle handle
destroy table [family] handle handle
```

## 2.3 Chain

### Definition

- Chains are containers for rules. They exist in two kinds, base chains and regular chains.
- A base chain is an entry point for packets from the networking stack, a regular chain may be used as

jump target and is used for better rule organization.
- For base chains, **type**, **hook** and **priority** parameters are mandatory.

| Type | Families | Hooks | Description |
|---|---|---|---|
| filter | all | all | Standard chain type to use in doubt. |
| nat | ip, ip6, inet | prerouting, input, output, postrouting | Chains of this type perform Native Address Translation based on conntrack entries. Only the first packet of a connection actually traverses this chain - its rules usually define details of the created conntrack entry (NAT statements for instance). |
| route | ip, ip6 | output | If a packet has traversed a chain of this type and is about to be accepted, a new route lookup is performed if relevant parts of the IP header have changed. This allows one to e.g. implement policy routing selectors in nftables. |

| Hook | Families | Description |
|---|---|---|
| prerouting | ip, ipv6, inet, bridge | All packets entering the system are processed by the prerouting hook. It is invoked before the routing process and is used for early filtering or changing packet attributes that affect routing. |
| input | ip, ipv6, inet, bridge, arp | Packets delivered to the local system are processed by the input hook. |
| forward | ip, ipv6, inet, bridge | Packets forwarded to a different host are processed by the forward hook. |
| output | ip, ipv6, inet, bridge, arp | Packets sent by local processes are processed by the output hook. |
| postrouting | ip, ipv6, inet, bridge | All packets leaving the system are processed by the postrouting hook. |
| ingress | ip, ipv6, inet, bridge, netdev | All packets entering the system are processed by this hook. It is invoked before layer 3 protocol handlers, hence before the prerouting hook, and it can be used for filtering and policing. Ingress is only available for Inet family (since Linux kernel 5.10). |

| | | |
|---|---|---|
| egress | netdev | All packets leaving the system are processed by this hook. It is invoked after layer 3 protocol handlers and before **tc** egress. It can be used for late filtering and policing. |

- Standard priority names, family and hook compatibility matrix

| Name | Value | Families | Hooks |
|---|---|---|---|
| raw | -300 | ip, ip6, inet | all |
| mangle | -150 | ip, ip6, inet | all |
| dstnat | -100 | ip, ip6, inet | prerouting |
| filter | 0 | ip, ip6, inet, arp, netdev | all |
| security | 50 | ip, ip6, inet | all |
| srcnat | 100 | ip, ip6, inet | postrouting |

- Standard priority names and hook compatibility for the bridge family

| Name | Value | Hooks |
|---|---|---|
| dstnat | -300 | prerouting |
| filter | -200 | all |
| out | 100 | output |
| srcnat | 300 | postrouting |

## Command

```
{add | create} chain [family] table chain [{ type type hook hook [device device]
priority priority ; [policy policy ;] [comment comment ;] }]
```

```
{delete | destroy | list | flush} chain ['family] table chain
list chains [family]
delete chain [family] table handle handle
destroy chain [family] table handle handle
rename chain [family] table chain newname
```

## 2.4 Rule

### Command

```
{add | insert} rule [family] table chain [handle handle | index index] statement
... [comment comment]
replace rule [family] table chain handle handle statement ... [comment comment]
{delete | reset} rule [family] table chain handle handle
destroy rule [family] table chain handle handle
reset rules [family]
reset rules table [family] table
reset rules chain [family] table [chain]
```

### 2.5 Set

### Definition

- Set consist of one or more elements, separated by commas, enclosed by curly brace.
- Anonymous sets are embedded in rules and cannot be updated, you must delete and re-add the rule.
- Named sets can be updated, and can be typed and flagged.

| Keyword | Description | Type |
|---|---|---|
| type | data type of set elements | string: ipv4_addr, ipv6_addr, ether_addr, inet_proto, inet_service, mark |
| typeof | data type of set element | expression to derive the data type from |
| flags | set flags | string: constant, dynamic, interval, timeout |
| timeout | time an element stays in the set, mandatory if set is added to from the packet path (ruleset) | string, decimal followed by unit. Units are: d, h, m, s |
| gc-interval | garbage collection interval, only available when timeout or flag timeout are active | string, decimal followed by unit. Units are: d, h, m, s |

| | | |
|---|---|---|
| elements | elements contained by the set | set data type |
| size | maximum number of elements in the set, mandatory if set is added to from the packet path (ruleset) | unsigned integer (64 bit) |
| policy | set policy | string: performance [default], memory |
| auto-merge | automatic merge of adjacent/overlapping set elements (only for interval sets) | |

### Command

```
add set [family] table set { type type | typeof expression ; [flags flags ;]
[timeout timeout ;] [gc-interval gc-interval ;] [elements = { element[, ...] } ;]
        [size size ;] [comment comment ;] [policy 'policy ;] [auto-merge ;] }
{delete | destroy | list | flush} set [family] table set
list sets [family]
delete set [family] table handle handle
{add | delete | destroy } element [family] table set { element[, ...] }
```

## 2.6 Map

### Definition

- Maps store data based on some specific key used as input. They are uniquely identified by a user-defined name and attached to tables.

### Command

```
add map [family] table map { type type | typeof expression [flags flags ;]
[elements = { element[, ...] } ;] [size size ;] [comment comment ;] [policy
'policy ;] }
{delete | destroy | list | flush} map [family] table map
list maps [family]
```

# B. Examples

### Filter

#### Filter config

```
#!/usr/sbin/nft -f
flush ruleset

table ip myip {
        chain web {
         tcp dport http accept
        tcp dport 8080 accept
    }
        chain myinput {
                type filter hook input priority 0; policy drop;
                ip saddr 192.168.56.112 accept
                ip saddr 10.0.3.0/24 jump web
        }
}

table bridge mybridge {
        chain myoutput {
                type filter hook output priority 0; policy drop;
                ether saddr 08:00:27:ef:94:c3 accept
        }
}

table ip mynat {
  chain myprerouting {
    type nat hook prerouting priority dstnat;
    tcp dport { ssh, http } dnat to 10.0.3.1
  }
  chain mypostrouting {
    type nat hook postrouting priority srcnat;
    ip daddr 10.0.3.1 masquerade
  }
}
```

### Rule

#### Rule command

```
$ nft add table ip myip
$ nft add chain ip myip myinput { type filter hook input priority 0; policy drop;
}
$ nft add rule ip myip myinput ip saddr 192.168.56.112 accept
$ nft add rule ip nat prerouting dnat tcp dport map { 80 : 192.168.1.100, 8888 :
192.168.1.101 }
```

# 1. Moving from iptables to nftables

- Translate by command

```
$ iptables-translate -A INPUT -p tcp --dport 22 -j ACCEPT
nft add rule ip filter INPUT tcp dport 22 counter accept
```

- Translate by whole ruleset

```
$ iptables-save > save.txt
$ cat save.txt
# Generated by iptables-save v1.6.0 on Sat Dec 24 14:26:40 2016
*filter
:INPUT ACCEPT [5166:1752111]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [5058:628693]
```

```
-A FORWARD -p tcp -m tcp --dport 22 -j ACCEPT
COMMIT
# Completed on Sat Dec 24 14:26:40 2016
$ iptables-restore-translate -f save.txt
# Translated by iptables-restore-translate v1.6.0 on Sat Dec 24 14:26:59 2016
add table ip filter
add chain ip filter INPUT { type filter hook input priority 0; }
add chain ip filter FORWARD { type filter hook forward priority 0; }
add chain ip filter OUTPUT { type filter hook output priority 0; }
add rule ip filter FORWARD tcp dport 22 counter accept
```

# C. References

| No | Info | Link/file/name of ebook |
|---|---|---|
| 1 | nfttables definition | https://wiki.archlinux.org/title/nftables |