nftables

Kadlecsik József
MTA Wigner FK
kadlec@blackhole.kfki.hu

Iptables

- 1999 óta: Linux csomagszűrő tűzfal
- Valójában két (három) részből áll:
  - Netfilter framework a Linux kernelben
    - iptables csomagszűrő tűzfal
  - iptables usertérbeli program a konfiguráláshoz

Több rendszer

- iptables
- ip6tables
- arptables
- ebtables

- Connection tracking
- Naplózás
- Queuing

Sikeres rendszer

- A netfilter keretrendszer változatlan 1999 óta!
- Rugalmas, moduláris
- Barátságos fejlesztői és user community

De...

- Kód sokszorozódás
  - ebtables
- Rugalmatlan kernel-userspace kommunikáció
  - Új verziók
  - Hibaüzenetek
- Dinamikus szabály változtatás
- Linearitás
  - ipset

Új rendszer: nftables

- Új szintaxisú leíró nyelv
  - Bison parser, nem getopt
- Egyetlen, protokoll-független rendszer
  - nft
- Gyorsabb feldolgozás: set, map, dictionary
- Jobb dinamikus szabály-feldogozás
- Publikus API
- 3.13-tól a vanilla kernelben

Változatlan

- Netfilter framework
- Connection tracking
- Naplózó interfészek
- Queuing

Szintaxis

- # komment
- Folytatósor: \
- Több parancs elválasztása: ;
- include *filename*
- Változó definiálás

```
define ext_if = eth0
define int_if = eth1
define all_if = { $ext_if, $int_if }
```

Táblák

- Nincsenek előre definiált táblák
- Tábla családok vannak:
  - ip, ip6, arp, bridge
  - inet
  - netdev
- Minden tábla default üres

Tábla példák

```
# nft add table ip filter
# nft add table ip6 filter
# nft list tables
table ip filter
table ip6 filter
# nft list table ip filter
table ip filter {
}
```

Láncok I.

- Alap láncok: név, típus, hook, prioritás
  - Lánc típus
    - filter, nat
    - route
  - Hook
    - prerouting, input, forward, output, postrouting
    - ingress (netdev)

Láncok II.

- Alap láncok, prioritás

NF_IP_PRI_CONNTRACK_DEFRAG	-400
NF_IP_PRI_RAW	-300
NF_IP_PRI_SELINUX_FIRST	-225
NF_IP_PRI_CONNTRACK	-200
NF_IP_PRI_MANGLE	-150
NF_IP_PRI_NAT_DST	-100
NF_IP_PRI_FILTER	0
NF_IP_PRI_SECURITY	50
NF_IP_PRI_NAT_SRC	100
NF_IP_PRI_SELINUX_LAST	225
NF_IP_PRI_CONNTRACK_HELPER	300

- Nem alap láncok

Láncok II.

```
# nft add chain [ip] filter input \
  { type filter hook input priority 0\;
    policy accept\; }
# nft add chain [ip] filter mychain
# nft add chain [ip] filter snatlog \
  { type filter hook postrouting \
        priority 101\; }
```

Szabályok

- Nincs match és target
- Expressions és statements
  - Több “target” egy szabályban

Kifejezések

- Kifejezések

```
adat operátor érték
```

  - Operátorok:
    - Hiányzik vagy == (eq)
    - != (ne)
    - <, <= (lt, le)
    - >, >= (gt, ge)
  - Bináris operátorok
    - &, |

Adatok

- Adat típusok: integer, bitmask, string, link layer, IPv4, IPv6 cím
- Payload adatok:
  - ether: saddr, daddr, ethertype
  - vlan: id, cfi (Canonical Format Indicator), ...
  - arp: htype, ptype, ...
  - ip: saddr, daddr, protocol, hdrlength, length, tos, ttl, ...

Adatok II.

- Payload adatok:
  - ip6: saddr, daddr, nexthdr, length, hoplimit, ..
  - tcp: sport, dport, flags, sequence, ackseq, ...
  - udp: sport, dport, length, checksum
    - udplite, sctp, ddcp, ah, esp
  - ct: state, direction, mark, expiration, ...
- Meta adatok: length, priority, mark, iif, oif, iifname, oifname, skuid, skgid, rtclassid

Állítások

- Terminál
  - accept, drop, queue, continue, return, jump|goto *chain*
  - reject, nat, queue
- Nem terminál
  - log, limit, counter, meta

Szabályok kezelése

```
# nft add|insert rule filter output \
    ip daddr 8.8.8.8 counter
# nft list -n table filter
table ip filter {
 chain output {
  type filter hook output priority 0;
  ip daddr 8.8.8.8 counter packets 0 .
 }
}
```

List, export, import

```
# nft list ruleset [arp|ip|ip6..]
# nft -f table-file

# nft export xml|json
```

Atomi szabály helyettesítés

- A tábla fájl(ok)ban van (filter-table)

```
flush table ip filter

table ip filter {

..

}

# nft -f filter-table
```

- Szükséges a flush parancs

Sorrendiség

- Két nem azonos szabály

```
# nft add rule filter input \
    ip protocol tcp counter
# nft add rule filter input \
    counter ip protocol tcp
```

TCP példák

```
# nft add rule filter input \
    tcp flags != syn counter
# nft add rule filter input \
    tcp flags & (syn | ack) == \
    (syn | ack) counter log
```

Meta példák

```
# nft add rule filter input \
   meta oifname lo accept
# nft add rule filter input \
   meta oif lo accept
# nft add rule filter input \
   meta mark 123 counter
# nft add rule filter output \
   meta skuid 1001 counter
```

Ct példák

```
# nft add rule filter input \
   ct state established,related accept
# nft add rule filter input \
   tcp dport 22 ct state new \
   log prefix \"New ssh\" accept
```

Intervallumok

- Kifejezések *tól-ig* formátumban

```
# nft add rule filter input \
    ip daddr 10.1.1.1-10.1.1.28 \
    drop
# nft add rule filter input \
    tcp ports 1-1024 drop
```

Sets

- Anonim set

```
# nft add rule filter output \
    tcp dport { 22, 443 } accept
```

- Adott szabályhoz kötött, nem lehet módosítani

Sets II.

- Névvel rendelkező set:
  - Adott táblához kötött, módosítható

```
# nft add set filter banned \
    { type ipv4_addr \; }
# nft add element filter banned \
    { 192.168.1.1-192.168.1.11, \
      192.168.2.34 }
# nft add rule filter input \
    ip saddr @banned drop
```

Sets III.

- Támogatott adat típusok
  - ipv4_addr
  - ipv6_addr
  - ether_addr
  - inet_proto
  - inet_service
  - mark

Maps

- Elemekhez elemeket rendelünk
- Literal

```
# nft add rule ip nat prerouting \
  dnat tcp dport map { \
    80 : 192.168.1.1, \
    443 : 192.168.2.2 }
```

Maps II.

- Deklarált

```
# nft add map nat port2ip \
  { type inet_service : ipv4_addr \; }
# nft add element nat port2ip \
  { 80 : 192.168.1.1, \
    443 : 192.168.2.2 }
# nft add rule nat postrouting \
  snat tcp dport map @port2ip
```

Dictionaries

- Elemekhez rendelt döntések halmaza
- Literal

```
# nft add rule ip filter input \
  ip protocol vmap { \
    tcp : jump tcp-chain, \
    udp : jump udp-chain, \
    icmp : jump icmp-chain }
```

Dictionaries II.

- Deklarált

```
# nft add map filter mydict \
  { type ipv4_addr : verdict \; }
# nft add element filter mydict \
  { 192.168.1.1 : accept, \
    192.168.1.2 : drop }
# nft add rule filter input \
  ip saddr vmap @mydict
```

Concatenations

- Két vagy több elem összekapcsolása
- Literal

```
# nft add rule ip filter input \
  ip saddr . ip daddr . ip protocol \
   { 1.1.1.1 . 2.2.2.2 . tcp, \
     3.3.3.3 . 4.4.4.4 . udp } \
    counter accept
```

Concatenations II.

- Deklarált, dictionary

```
# nft add map filter services \
  { type ipv4_addr . inet_service : verdict \;
}
# nft add element filter services \
  { 192.168.1.1 . 22 : accept, \
    192.168.1.2 . 80 : drop }
# nft add rule filter forward \
  ip daddr . tcp dport vmap @services
```

Concatenations III.

- Deklarált, map

```
# nft add map nat natmap \
 { type ipv4_addr . inet_service :ipv4_addr \; }
# nft add element nat natmap \
  { 1.1.1.1 . 80 : 192.168.1.1, \
    1.1.1.2 . 8080 : 192.168.1.2 }
# nft add rule nat prerouting \
  dnat ip saddr . tcp dport map @natmap
```

A forrás

- Linux kernel 3.13 fölött
- git:
  - git://git.netfilter.org/nftables
  - git://git.netfilter.org/libnftnl
  - git://git.netfilter.org/libmnl

Goodies

- Nftables forráskönyvtár:
  - doc/
  - files/nftables/
  - files/nftables/examples/
- wiki.nftables.org

Kérdések?