Netfilter Mini Workshop

Florian Westphal
4096R/AD5FF600 fw@strlen.de
80A9 20C5 B203 E069 F586
AE9F 7091 A8D9 AD5F F600

Red Hat

netdev 0x15, Virtual

Agenda

Updates to netfilter components since last netdev conf
- iptables and ipset
- connection tracking and ipvs
- flowtable
- nftables

work in progress
- recent changes

iptables and ipset updates

- iptables
  - Two releases: 1.8.6, 1.8.7
  - mostly bug fixes
  - lock file name can be overridden via environment variable
  - iptables-nft -L no longer creates an empty ruleset
- ipset
  - Five releases: 7.7-7.11
  - new "bucketsize" option to limit hash chain lengths

connection tracking and ipvs updates

- conntrack
  - sctp: allow heartbeat after connection re-use
  - ctnetlink: expose timeout and protoinfo in destroy events
  - add sysctl for tcp timeout pickup after offload
- ipvs
  - Weighted twos choice LB algorithm
  - fix 1 second delay on connection reuse

flowtable updates

- split replace, destroy and stats to different workqueues
- add support for different xmit types
  - old: neigh, xfrm (ipsec)
  - new addition: direct xmit
    - used when HW offload is enabled
    - used for bridge
- bridge vlan filtering
- VLAN, DSA and PPPoE support

nftables updates

- Three releases: 0.9.7, 0.9.8, 0.9.9
- bug fixes and new features
- improves load/list time for rulesets with large number of chains
- 20 contributors

implicit chains

Allows to group rules in a subchain:

```
table inet x {
  chain y {
    type filter hook input priority 0;
    tcp dport 22 jump {
       ip saddr { 127.0.0.0/8, 172.23.0.0/16, 192.168.13.0/24 } accept
       ip6 saddr ::1/128 accept;
    }
  }
}
```

flag types are now listed correctly

nft can now reconstruct base values in sets that contain binary operations:

```
table ip test {
 set tcp_good_flags {
  type tcp_flag
  flags constant
  elements = { fin | psh | ack | urg, fin | psh | ack,
               fin | ack | urg, fin | ack, syn | psh | ack | urg,
               syn | psh | ack,
               syn | ack | urg,
               syn | ack, syn, rst | psh | ack | urg,
               rst | psh | ack, rst | ack | urg, rst | ack,
               rst, psh | ack | urg,
               psh | ack, ack | urg, ack }
   }
}
```

extended comment support

```
table ip x {
  comment "test"
  chain c {
    comment "test"
    type filter hook input priority 0;
  }
  set s {
    type ipv4_addr;
    comment "some_addrs"
    elements = { 1.1.1.1, 1.2.3.4 }
  }
}
```

This also works for maps and object definitions such as quotas or named counters.
attaching comments to rules or set elements was already supported.

tproxy socket wildcard match support

- iptables has a "socket" match
- it will not match sockets bound to the any address (::, 0.0.0.0) by default
- has a poorly-named "–nowildcard" flag to avoid the extra check
- nft "socket" expression did not support this
- can now use "socket wildcard 1" to match any-bound sockets

improved error reporting

- since 5.9 kernel reports the location of the expression that triggers a validation error
- nft now makes use of this:

```
# nft add rule x y jump test
Error: Could not process rule: No such file or directory
add rule x y jump test
                ^^^^
# nft 'add chain x y { type nat hook prerouting priority dstnat; }'
Error: Could not process rule: No such file or directory
add chain x y { type nat hook prerouting priority dstnat; }
                    ^^^
```

payload expression

- sctp checksum fixup. This makes stateless dnat work for sctp:
  `sctp dport set 32`
- can use multiple protocols per nat rule, e.g.:
  `meta l4proto { tcp, udp } dnat ip to 1.1.1.1:80`
- raw tcp option match support: (kind,offset,length) `tcp option @42,4,2`
  `tcp option 42 exists`
- sctp chunk matching `sctp chunk { data, init , .. } exist`
  `sctp chunk asconf seqno 12345`
- QinQ support

```
ether type 8021ad vlan id 1 vlan type 8021q \
                  vlan id 2 vlan type ip counter
```

cgroupv2 support

cgroupv2 path is expressed relative to /sys/fs/cgroup

```
# nft add rule x y socket cgroupv2 level 1 "user.slice" counter
# nft list ruleset
table ip x {
  chain y {
     type filter hook input priority filter; policy accept;
     socket cgroupv2 level 1 "user.slice" counter
  }
}
```

Set element multi-statement support (1)

associate element with both counter and limit

```
set y {
  type ipv4_addr
  limit rate 1/second counter
  elements = { 5.5.5.5 limit rate 1/second counter packets 0 bytes 0 }
}
```

or add elements later on:

```
nft add element x y { 1.1.1.1 limit rate 1/second counter }
```

Set element multi-statement support (2)

Set can also be added to from the packet path

```
set y {
  type ipv4_addr
  size 65535
  flags dynamic,timeout
  timeout 1h
}
chain z {
  type filter hook forward priority 0;
  update @y { ip daddr limit rate 1/second counter }
}
```

Set element catch-all support

Can add '*' element to assign a default value

```
table x {
 set y {
  type ipv4_addr
   counter
   elements = { 10.2.3.4 counter packets 1 bytes 112,
                * counter packet 12 bytes 820 }
  }
}
```

can assign a default return value in maps

table owner flag support

Can request exclusive table ownership

- Only owning program can make changes
- auto-removed when program exits

```
# nft -i
nft> add table ip x { flags owner; }
nft> list ruleset
table ip x { # progname nft
  flags owner
}
```

Unreleased / work in progress

features that are not (yet) part of a release:

- base hook listing
- "last" expression

base hook listing

list registered base hooks

```
# nft list hooks ip output
family ip hook output {
  -000400 ipv4_conntrack_defrag
  -000200 ipv4_conntrack_local
  -000150 iptable_mangle_hook
  -000100 nf_nat_ipv4_local_fn
  +000000 iptable_filter_hook
}
```

This exposes the complete pipeline, including implicit functions e.g. due to connection tracking.

base hook listing

list registered base hooks

```
# nft list hooks ip output
family ip hook output {
  -000400 ipv4_conntrack_defrag
  -000200 ipv4_conntrack_local
  -000150 iptable_mangle_hook
  -000100 nf_nat_ipv4_local_fn
  +000000 iptable_filter_hook
}
```

This exposes the complete pipeline, including implicit functions e.g. due to connection tracking.

last expression

`last` records timestamp when rule is matched

```
nft list ruleset
[..]
tcp dport 22 accept last used 2m8s28m
```