
# nftables: a new packet filtering framework for Netfilter

pablo@netfilter.org

OSD 2013

Copenhague, Denmark

# nftables: Intro

- New kernel packet filtering framework to replace iptables.
- No changes in the core infrastructure:
  - Netfilter hooks
  - Connection Tracking System
  - NAT
- Designed from lessons learnt from iptables.
- Provides backward compatibility infrastructure.
- Nftables released in March 2009 by Patrick McHardy.
- Currently under active development.

# nftables vs. iptables: Architecture

- Pseudo-state machine in kernel-space (similar to BPF).
- Registers: 4 general purpose (128 bits long each) + 1 verdict.
- Provides instruction set (can be extended):
  - reg = pkt.payload[offset, len]
  - reg = immediate(value, len)
  - reg = cmp(reg1, reg2, EQ)
  - reg = byteorder(reg1, NTOH)
  - reg = pkt.meta(mark)
  - reg = (reg1 & mask) ^ xor)
  - reg = lookup(set, reg1)
  - reg = ct(reg1, state)
  - reg = lookup(set, data)
- New extensions are implemented using this instruction set.
- Netlink interface: kernel ↔ userspace (http://1984.lsi.us.es/~pablo/docs/spae.pdf)

# nftables vs. iptables: Architecture

- Extensions: Matches and targets
- New extensions are written in C:
  - 1 Linux kernel module: xt_blah.c
  - 1 libxt_blah.c file under user-space iptables tree.
- Binary array containing the rule-set.
- communication kernel ↔ userspace: Use setsockopt()/getsockopt()
- Poor incremental dynamic rule-set updates
- Limitations:
  - Extending existing extensions

# nftables vs. iptables: Rule handling

- Adding rule: match ip saddr 1.1.1.1 tcp dport 80, accept:
  - Step 1: Parse command line
  - Step 2: Build rule from user-space using instruction set:
    - Reg1 = pkt.payload(offset(ip saddr), 4)
    - Reg2 = immediate(1.1.1.1, 4)
    - RegV = cmp(reg1, reg2, EQ) // implicit return if mismatch
    - Reg1 = pkt.payload(offset(tcp dport), 2)
    - Reg2 = immediate(80, 2)
    - RegV = immediate(DROP)
  - Step 3: Convert that to netlink and pass message with code to kernel.
- Deleting rule
  - Step 1: Dump rule-set (to check which one to delete)
  - Step 2: Delete by rule identifier

# nftables vs. iptables: Rule handling

- Adding rule: Match -s 1.1.1.1 -p tcp –dport 80, accept:
  - Step 1: Parse command line
  - Step 2: Build rule match and target: use built-in source, tcp match and standard target
  - Step 3: Get rule-set from kernel (binary), update it with rule
  - Step 4: Pass rule-set to kernel space via setsockopt()
- Delete rule:
  - Step 1: Parse command line
  - Step 2: Convert rule to binary.
  - Step 3: Dump existing ruleset (in binary).
  - Step 4: Find rule matching in ruleset (binary comparisons)
  - Step 5: If found. Allocate new rule-set, build it and pass it to kernel-space.

# Nftables vs. Iptables from developer

- Iptables provided no third party library
- Libipt/libipt6 probably, you have to work with binary blobs
- Nftables provides libnftables and will provide high level library to work in an object oriented fashion.

# nftables from userspace

- **Backward** compatible:
  - Utility derivated from iptables/ip6tables with same syntax.
  - You can use all existing xtables modules.
  - You can still add new xtables extensions in the same fashion.
  - No need to learn new utilities if you don't want to.
  - No need for new documentation.
  - No need to update your scripts.
- **But also, new features without breaking backward compatibility**:
  - xtables-event : Reporting changes in tables/chains/rules
  - Better incremental rule update support: Matches internal state is not lost
  - Enable/disable the chains per table that you want
  - … more improvements for xtables yet to come

# nftables from userspace

- New utility nft (still under work):
  - New syntax, new features.
- Fast lookups:
  - tcp dport { 80 => accept, 22 => drop }
  - ip daddr {
    192.168.0.0/24 => jump chain1,
    192.168.1.0/24 => jump chain2,
    }
  - ip saddr . tcp dport {
    1.1.1.1 . 80 => accept,
    1.1.1.2 . 22 => drop,
    }

# nftables summary

- One single kernel framework for packet filtering allowing long term evolution.
- Two userspace tools:
  - Backward compatible utility:
    - Same syntax + same features + new features
  - New utility:
    - New syntax + more cool new features
- Still work in progress.
- There will be user-friendly documentation.

# Nftables summary (2)

- Grab the code
  - Backward compatible utility:
    - Kernel: git://1984.lsi.us.es/nftables
    - Library: git://1984.lsi.us.es/libnftables
    - User-space: git://1984.lsi.us.es/iptables-nftables
  - New utility:
    - Library: git://git.netfilter.org/libnl-nft
    - User-space: git://git.netfilter.org/nftables


# nftables: a new packet filtering framework for Netfilter

pablo@netfilter.org

OSD 2013

Copenhague, Denmark