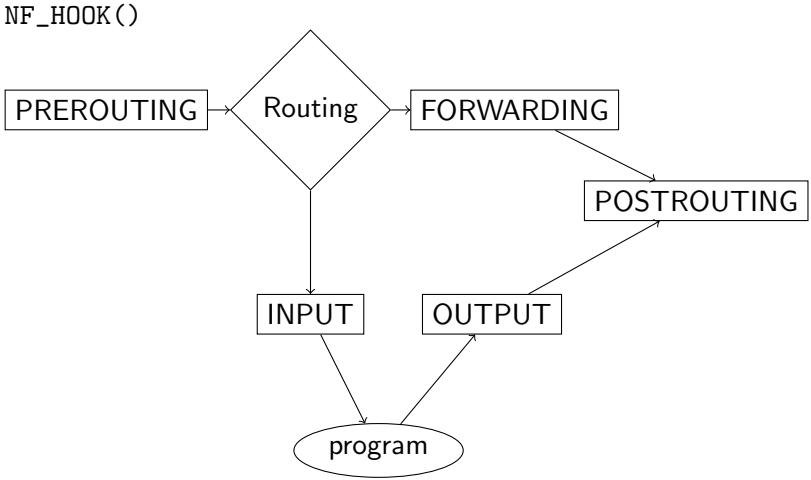
recent and ongoing netfilter work

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

Red Hat

netdev 0x13, Prague

High-level netfilter overview



jump labels to keep overhead close to 0 if unused

connection tracking uses almost all hook locations

single `iptables -A FORWARD -m conntrack ...`: indirect calls from base hooks alone:

1. defrag in (prerouting)
2. conntrack in (prerouting)
3. filter table (input)
4. conntrack confirm/helper (input)
5. filter table (forward)
6. defrag out (output)
7. filter table (output)
8. conntrack local (output)
9. conntrack confirm/helper (postrouting)

4 base hooks (indirect calls) for each packet (4.20: 5)
nat, mangle and/or raw table(s) bring in even more
necessary evil: usually called functions reside in kernel modules

indirect calls are a problem nowadays

indirect calls are expensive – adding `NOTRACK` rules has almost no noticeable effect anymore in some benchmarks:

```
iptables -t raw -p tcp --dport 12345 -j NOTRACK
```

- one indirect call for raw prerouting, one for output
- one indirect call for tcp port match
- one indirect call for NOTRACK target

conntrack packet path is (now) free of indirect calls
nf_nat packet path work almost done by now
indirections remaining for `nf_conn` destruction: wip

(Mostly) useless indirection: defrag hook

- most packets are not ip(v6) fragments
- we eat indirect call cost, only to return immediately in almost all cases
- merge back into conntrack hooks?
  - impacts raw table functionality, probably not doable without breaking existing setups
- Could attempt to annotate defrag hook and do the "is fragment" check in the netfilter core
  - not nice from a design (layering) point of view
  - might be worth trying to see how ugly this would look like (ipv6!)

nf tables

- terminology: "expression" is nft kernel equivalent of iptables matches and/or target
- functionality of xtables modules is usually replicated by combining several expressions, e.g. meta + cmp or payload + range
- some expressions are handled directly in evaluation loop: not even a direct function call done
- all built-in expressions (cmp, payload, meta, ...) are called directly
- indirections only for those that are modular
  - make more built-in? Several candidates exist, e.g. `counter`
    - set infrastructure should probably be made built-in too
    - some are ok as-is, e.g. `nft_log` (not hot path)
  - could add small built-in replica of modular ones, e.g. `nft_ct` version that can only handle `ct status`.

nf tables (2)

- NAT support for the 'inet' family almost complete, nat ipv4/ipv6 modules are gone
  - protocol trackers merged with nf nat core
  - ipv6 dependencies handled via indirect call (`CONFIG_IPV6=m`) or direct one (`CONFIG_IPV6=y`)
- no need to add two nat tables for simple "`oif ethX masquerade`" anymore
- for dnat/snat, new syntax:

```
table inet nat { [ .. ]
  # detect af from network protocol context:
  ip6 daddr dead::2::1 dnat to dead:2::99
  # use new dnat ip6 keyword:
  dnat ip6 to dead:2::99
       ^^^
}
```

nf tables (3)

broute support (select packet for routing rather than bridging)

- requires refactoring to make ebtables broute work via normal hook infra
- current broute hook will be removed
- ebtables broute table will continue to work as-is
- nft will use explicit broute expression (ebtables overloads `DROP`)