

bridge filtering with nftables

Florian Westphal

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

Red Hat

February 2016

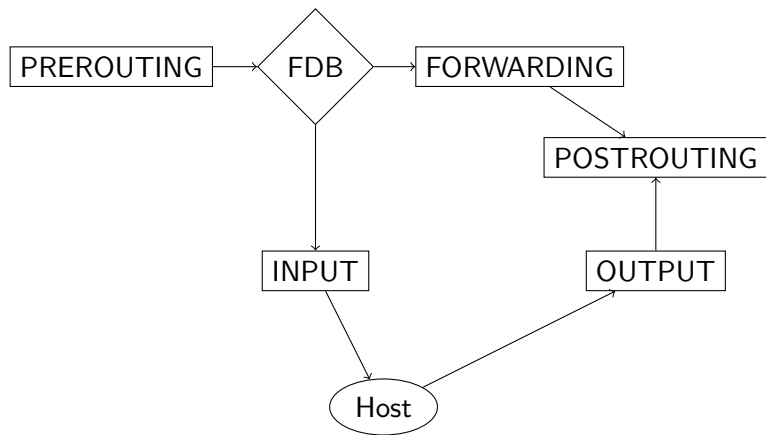
Intro

- ▶ same concept as netfilter ipv4/ipv6
- ▶ “hooks” are placed at various spots in the bridge module
- ▶ kernel modules can register functions to be called at these hook points

```
NF_HOOK(NFPROTO_BRIDGE, NF_BR_PRE_ROUTING, skb, skb->dev,  
        NULL, br_handle_frame_finish);
```

- ▶ iterates over registered functions and calls them
- ▶ functions can decide fate of packet (continue, drop, ...)

Kernel Bridge Hooks



current state: ebttables

- ▶ cloned off iptables more than a decade ago
- ▶ offers a few (stateless) matches and targets
 - ▶ ip/ip6 addresses, vlan id
 - ▶ packet type (i.e. multicast, broadcast, ...)
 - ▶ packet mark
 - ▶ src/dst MAC rewrite, redirect to local stack support (routing, proxies)
- ▶ bridge netfilter hooks placed in bridge code, similar to ip stack
- ▶ no stateful matches, no connection tracking
- ▶ ebttables cannot use xtables targets or modules

call-iptables (1)

- ▶ `br_netfilter.ko` – implements `call-iptables` mode:
- ▶ invoke `ip/ipv6 nf` hooks from the bridge path
- ▶ upside:
 - ▶ provides all `xtables` modules and targets (via `iptables` ruleset on the bridge)
 - ▶ `conntrack` support, L3/L4 NAT
- ▶ downside:
 - ▶ many subtle layering violations and problems
 - ▶ `inet` “owns” `skb->cb[]`: save/restore for each `iptables` trip
 - ▶ in `iptables` `indev` and `outdev` is bridge, i.e. `-i br0` instead of bridge port
 - ▶ VLAN mess: allows temporary removal of VLAN header
 - ▶ end host/router doesn't care, filtering via `ifname` (`"-i eth0.42"`)
 - ▶ VLAN data only accessible from bridge hooks

call-iptables (2) – conntrack

- ▶ the good news: It'll work
- ▶ unless you have overlapping addresses (different bridges, VLANs)
- ▶ unless you need/use NFQUEUE to have packets inspected by userspace (e.g. suricata).
- ▶ bridge has to consult ipv4 FIB to cope with NAT
- ▶ also needs to cope with e.g. re-lookup of the destination MAC address
- ▶ `skb->nf_bridge` exists only because of the call-iptables mode

Summary

- ▶ bridge filtering is usually done with both ebttables and iptables rulesets
- ▶ ... because thats the only way you get conntrack + iptables features
- ▶ at least one long standing oops (call-iptables+conntrack+nfqueue) remaining
- ▶ another problem: netfilter hooks are per namespace, not per bridge
- ▶ ebttables == 'iptables from 2001' (e.g. rwlock in main traverser)

nfqueue

- ▶ pass packets to userspace via netlink
- ▶ userspace can drop or accept packet
- ▶ userspace can rewrite/replace payload
- ▶ backend is limited to ip and ipv6
- ▶ netlink attributes provided (incomplete list):
 - ▶ the family and hook that queued the packet
 - ▶ in and outgoing interface index
 - ▶ packet nfmark
 - ▶ packet payload (starts w. network header – `skb->data`)

nfqueue for bridge

- ▶ most simple solution: just push/pull eth header
- ▶ i.e. payload attribute starts with ethernet header
- ▶ several issues with this approach
 - ▶ VLAN untag or extra attribute?
 - ▶ how to allow l2 header rewriting?
Doesn't really work since we might have to pull more data, e.g. added qinq
- ▶ seems preferable to add new netlink attributes
- ▶ bonus: can use this for netdev family too

nfqueue for bridge, plan

- ▶ add new attributes for L2 and VLAN header
- ▶ L2 – `skb_mac_header()`
 - ▶ can add L2 header expansion/shrinking since attr size is known
- ▶ VLAN – serialize skb vlan metadata
 - ▶ no need to untag queued skb
 - ▶ allows later addition of VLAN header removal

Handling overlapping addresses

- ▶ distinct VLANs might have duplicate addresses
- ▶ already a problem w. `call-iptables` and `bridge-nf-filter-vlan-tagged` enabled
- ▶ partial workaround for `conntrack` with `-j CT --zone x`, but not for `defrag`
- ▶ requires manual setup
- ▶ not very efficient with lots of vlan zone pairs due to `xtables` limitations

Handling overlapping addresses (2)

- ▶ could extract vlan id and use that as additional key
- ▶ would need kernel change when e.g. asking to conntrack ip inside PPPoE frames
- ▶ in light of this manual setup doesn't seem too bad

nft would not suffer from 'linear ruleset' xt problem, e.g. use map:

```
add rule bridge track pre ct zone set vlan id map {  
    1 : 1, 2 : 2 ...
```

bridge conntrack

plan: add new bridge ct expression

- ▶ serves as ingress hook point and conntrack-enable switch
- ▶ native bridge hooks for confirmation, helpers and reply direction
- ▶ look at `skb->protocol`, then do upcall to ipv4 or ipv6 ct handler
- ▶ no tracking for outgoing connections – ip(6) conntrack already does it

bridge conntrack hook overview

add following bridge hooks:

- ▶ postrouting:
 - ▶ helper
 - ▶ confirmation
- ▶ prerouting:
 - ▶ reply traffic handling, i.e. no NEW ctstate also handle IP(v6) defragmentation here
- ▶ absent:
 - ▶ no input confirm (would be handled by ipv4/ipv6 hook)
 - ▶ prerouting input for new connections – only via explicit rule

no auto-defrag?

- ▶ ipv4/ipv6 conntrack forces defrag via module dependency
- ▶ "How can I disable `nf_defrag_ipv4` on `ethX`"?
You can't
- ▶ How would one go about to allow this?
Best idea so far: defrag expression:
`add rule bridge track pre meta iif not ethX defrag`
- ▶ Need for manual config doesn't play nice with conntrack RELATED handling
- ▶ seems best to force-on once conntrack is used/activated

Future Work

- ▶ implement defrag+conntrack as outlined
- ▶ try to avoid dependencies – no ipv6 conntrack outload for example
- ▶ L3/L4 nat not planned at the moment
- ▶ can use ether set daddr plus pkttype set unicast to redirect packet to bridge for routing

Questions?