resilient rx & scaling udp

willemb@google.com

netconf 2019

resilient rx: context

---

fun input

    ipv4 options, ipv6 exthdr, icmp, tunnels, ...
    front-end / back-end

security@ / CVEs

    recent: tcp_gso_segs overflow

syzkaller

    recent: ipv4 options

Roll out across "fleet"

    borg, CDN, VMs, ..
    live patching

resilient rx: progress

bpf flow dissector

eth_get_headlen

configurable gro

sysctl bitmap, 256 IPPROTO_..

resilient rx: just add BPF

less intrusive, more programmable

drop: unexpected protocols + extensions, crafty packets
bypass gro: all but hot path

resilient rx: just add XDP

```
less intrusive, more programmable

drop: XDP_DROP
bypass gro
    skb->gro_bypass = 1 (or napi, PER_CPU rx_state, …)
```

resilient rx: just add XDP

```
less intrusive, more programmable

drop: XDP_DROP
bypass gro
   skb->gro_bypass = 1 (or napi, PER_CPU rx_state, …)

generic_xdp
   linearized skb
```

resilient rx: that's a lot of packet parsing

XDP: drop

BPF flow dissector: eth_get_headlen(**skb,** ..)

GRO

BPF flow dissector: RPS skb_get_hash

BPF tc ingress : tunnel decap

ip_rcv, tcp_v6_rcv, ..

resilient rx: that's a lot of packet parsing

XDP: drop

BPF flow dissector: eth_get_headlen(**skb,** ..)

GRO

BPF flow dissector: RPS skb_get_hash

BPF tc ingress : tunnel decap

ip_rcv, tcp_v6_rcv, ..

code path linearization

resilient rx: one parsing stage

set from BPF flow dissector (better: XDP, *how?*)

- XDP_DROP
- bypass_gro
- gro_flow_key
- hash, l4_hash, sw_hash

resilient rx: one parsing stage

---

set from BPF flow dissector (better: XDP, *but how*)

- XDP_DROP
- bypass_gro
- gro_flow_key
- hash, l4_hash, sw_hash

IMPL

- eth_get_headlen
- __skb_flow_bpf_to_target
- flow_keys_basic -> flow_keys
  - do not zero each time

resilient rx: one parsing stage ++

- XDP_DROP
- bypass_gro
- gro_key
- hash, l4_hash, sw_hash
- decap
- sk_lookup + XDP_REDIRECT to non-xsk sk?
    - if !ip->frag && !sk->has_frags_outstanding

resilient rx: one more thing..

"rx": also tx path, from VMs

syzbot ❤️ gso
(hint: this is not a good thing)

more VIRTIO_NET_HDR_GSO_.. types, like tunnels?

net: validate untrusted gso packets
gso: validate gso type in gso handlers [discussion]
net: validate untrusted gso packets without csum offload
gso: validate gso type on ipip style tunnels

GSO is not a validator
BPF flow dissect on input and deprecate SKB_GSO_DODGY?

udp scaling

use case: Youtube over QUIC

progress:

segmentation offload

GSO + LSO (mlx5, *intel, mlx4, ..*)

udp zerocopy tx

pacing offload: SO_TXTIME + FQ

udp gro, listified rx, udp gro frag_list

measured benefits

opportunity: scale gso 4 -> 45 mss, ACKs, crypto

udp scaling: loose ends

```
ipv6 flow label
   IPV6_FL_A_GET
   fl6_sock_lookup
ip_recverr MTU errors
unconnected sk_txhash SCM_HASH
timestamping
   first, last segment,
   xmit_more
      skb_tx_timestamp before ++prod_idx
         a few missing, incl. mlx5
```