Design XDP feature detection for less error prone User eXperience

Toke Høiland-Jørgensen (Red Hat)
Jesper Dangaard Brouer (Red Hat)

Netconf
Boston, June 2019

The XDP available features issue

Users cannot know if a device driver support XDP or not

- This is the most often asked question
- And people will often use XDP-generic without noticing,
  - and complain about performance... this is a support issue.

Real users requesting this:

- Suricata config want to query for XDP-support, else fallback to BPF-TC
- VM-migration want to query for XDP-support, else need to abort migration

Original argument: Drivers MUST support all XDP-features

- Thus, there is not reason to expose feature bits
- This was never true, and e.g. very few drivers support redirect

What is the real issue?

Simply exposing feature XDP to userspace, doesn't solve the real issue

- Real issue: too easy to misconfigure
- How to get users to check features before attach (API didn't prev exist)

Kernel allows users to attach XDP bpf_prog

- that use features that driver doesn't implement
- cause silent drops (debug via tracepoints)

We want something that can reject earlier

- at BPF-load or XDP-attach time

Original plan (didn't work): Hide feature check

Wanted to detect XDP features via analysing return codes used by program

- Not possible because map value (register) can be a return value
- Could still detect XDP_REDIRECT feature, as it needs a helper call

Explicit supply XDP features used, when attaching bpf_prog to a device

- Not possible due to BPF tail-call maps (`BPF_MAP_TYPE_PROG_ARRAY`)
- Driver attach `ndo_bpf` is bypassed, on map insert in tail-call map
- as map insert becomes second level XDP attach
- E.g. XDP using tail-calls, driver is unaware of map inserts
- E.g. verifying on attach, require traversing levels of tail-calls

One option left: Associate ifindex with bpf_prog at load time (like HW-offload)

- See later slide for challenges and pitfalls...

Internal kernel API

Should likely use net_device NDO (Network Device Operation) call `ndo_bpf`

- for query (or testing) supported XDP features
- API kept between driver and kernel-core

External userspace API

Add a userspace API to query features

- Netlink?
- Ethtool?

If BPF load time check is possible

- This could be the userspace API, which is a probe-API
- Similar to '`bpftool feature`' command that probe-loads BPF-progs

BPF-load time ifindex binding (possible???)

General idea (like HW-offload): Supply ifindex at BPF load time

Our old nemesis: tail-call maps

- Case#1: ifindex bound XDP-prog use tail-call (map)
  - Issue: what stops adding non-ifindex XDP-prog to this tail-call-map?
- Case#2: Can NIC-A XDP-prog tail-call XDP-prog bound to NIC-B?
- Solution(?): Bind tail-call map to ifindex? (on first insert)

Pitfall: Generic-XDP

- At BPF load time, don't know if used for native or generic-XDP
  - (1) supply more info than ifindex?
  - (2) let ifindex imply native-XDP?

Can verifier detect XDP features?

Either need to supply features (more input than `ifindex`)

- Or verifier needs to be able to detect features

Verifier detection strategy, to deduce XDP features in-use

- If XDP return code comes from register/map
  - then assume all XDP-return codes in use
- Except: can remove XDP_REDIRECT if redirect-helper isn't used

(Came up yesterday)

- verifier knows helpers used. e.g. `bpf_xdp_adjust_tail`
- With jumbo-frames, driver want to know if adjust-tail is used

End

Disclaimer

- These slides are only design ideas and suggestions
- Non of this is actually implemented

Main purpose was getting a discussion going

- which were hopefully successful...

Extra slides

Below extra slide

- with details if people want to discuss these

Code details: XDP-feature

Open Question: How do we express XDP-features?

- Drivers need to know about their features (simply bits? enough?)

Code details, extending the `ndo_bpf` commands:

```
struct netdev_bpf {  /* from: include/linux/netdevice.h */
        enum bpf_netdev_command command;
        union {
                /* XDP_QUERY_PROG, XDP_QUERY_PROG_HW */
                struct {
                        u32 prog_id;
                        /* flags with which program was installed */
                        u32 prog_flags;
                };
                /* XDP_QUERY_FEATURES */
                struct {
                        u64 flags;
                };
        [...]
        };
};
```