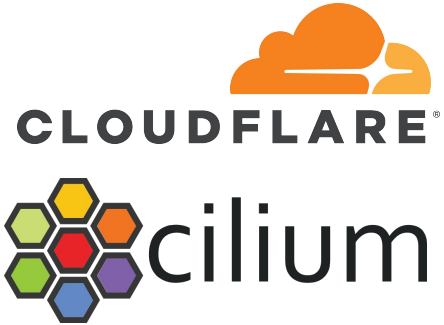


A pure Go eBPF library

Lorenz Bauer (Cloudflare), Joe Stringer (Cilium)

Our BPF use case

- Packet wrangling in XDP and TC
- Long-running service managing eBPF, written in Go

- Cloudflare: L4 load balancer
- Cilium: Container security for Kubernetes

CLOUDFLARE cilium

Available libraries

- **libbpf**: the canonical implementation
  Lives in the kernel repo; C
- **libbcc**: focused on tracing
  Wraps libbpf, LLVM

CLOUDFLARE cilium

libbcc

- Heavy runtime (LLVM dependency)
- Difficult to build and package
- github.com/iovisor/gobpf; uses CGo

libbpf

- Features land here
- Few external dependencies
- Relatively lightweight

No fully fledged Go wrapper

CLOUDFLARE cilium

The pure-Go syndrome

- Lots of rewriting non-Go libraries in Go
- github.com/vishvanda/netlink, ...

CLOUDFLARE cilium

Problems with CGo

- CGo calls are relatively expensive
  - ~10% overhead for a simple map_lookup_elem
- Bad developer experience
  - Link to library: OS packages, ABI, etc.
  - Copy source code: difficult to keep up-to-date

Problems with CGo contd.

- Makes tooling less useful
  - Cross-compilation
  - Debuggability
    - Profiling
    - Tracing

github.com/cilium/ebpf

- You guessed it: pure Go
- To write services managing eBPF
  - Load programs
  - Modify maps
  - Collect metrics, events, etc.
- MIT

Goals

- Cover networking use-cases
- Minimal external dependencies
- Well tested, highly testable
- Solve common problems

Non-goals

- Tracing: use libbcc
- Specific support for all hook points
  - Can live in separate libraries

Step 1: Map and Program

- Map
  - CRUD
  - Pinning
  - Misc: nested maps, per CPU array
- Program
  - Create and Pin

Step 2: Perf events

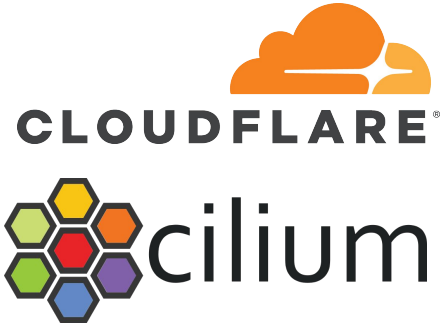
- Support for PERF_EVENT_ARRAY
- Probably as sub-package

In the future

- ELF loader
- BTF
- Global variables

Contributors contributors contributors!

- Does this sound useful?
- If not, why?



Questions?