# CFP-41634: Datapath Plugins

Jordan Rife

https://github.com/cilium/design-cfps/pull/76

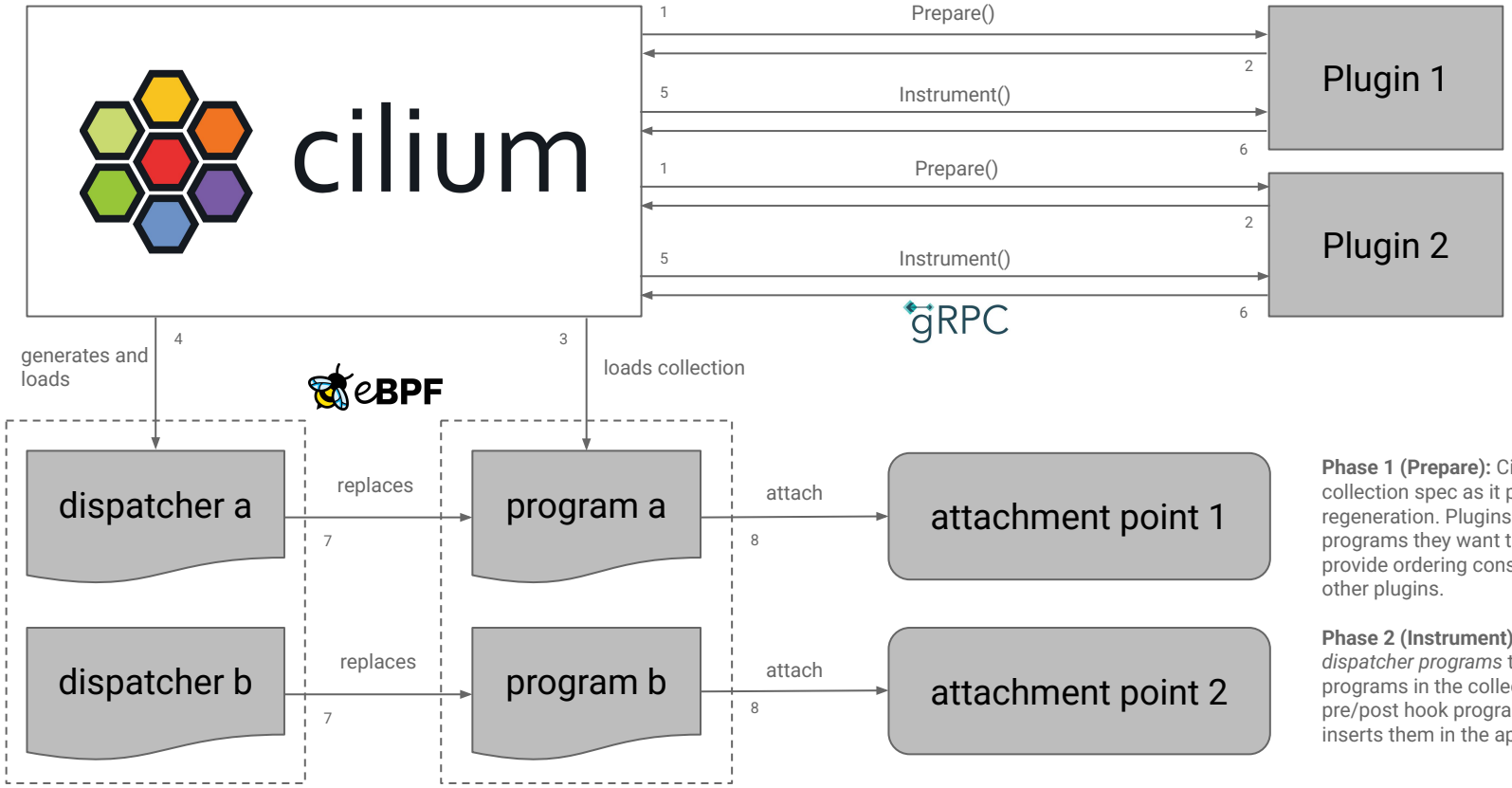
# Motivation

- Make it easier to combine Cilium with other internal components that themselves use BPF.
- Make it easier to implement custom dataplane features that integrate with Cilium rather than direct patching.
  - Custom transparent proxies
  - Custom observability
  - Custom encapsulation and routing

# What’s Different From Last Time?

- Last year we discussed something similar focused on structuring the datapath around well-defined extension points but there were concerns around maintenance burden and restrictions this imposes on the datapath in general.
- This design focuses mainly on the Cilium loader and aims to place little to no restrictions on the datapath itself.

# CFP-41634: Design Overview




**Phase 1 (Prepare):** Cilium loads a collection spec as it prepares for endpoint regeneration. Plugins tell Cilium which programs they want to instrument and provide ordering constraints relative to other plugins.

**Phase 2 (Instrument):** Cilium generates *dispatcher programs* that replace programs in the collection. Plugins load pre/post hook programs and Cilium inserts them in the appropriate place.

# CFP-41634: Dispatcher Mechanism

```
static int cilium_dispatch(const struct __ctx_buff *ctx) {
        int ret;

        ret = cilium_pre_1(ctx);
        if (ret != TC_ACT_UNSPEC)
                return ret;
        ret = cilium_pre_2(ctx);
        if (ret != TC_ACT_UNSPEC)
                return ret;
        ...
        ret = cilium(ctx);
        ret = cilium_post_1(ctx, ret);
        if (ret != TC_ACT_UNSPEC)
                return ret;
        ret = cilium_post_2(ctx, ret);
        if (ret != TC_ACT_UNSPEC)
                return ret;
        ...
        return ret;
}
```

https://docs.ebpf.io/linux/program-type/BPF_PROG_TYPE_EXT/

# CFP-41634: Multi Plugin Support

- Multiple plugins can exist side by side
- In the *prepare* phase, plugins provide Cilium with declarative ordering constraints for each hook.
  - `{ “anchor”: “before”, “plugin”: “some-other-plugin” }`
- Cilium ensures these constraints are respected when arranging program chains.

# Other Possibilities

- `fentry`/`fexit` for fine-grained runtime benchmarking or debugging.