AYA

Extending the Linux Kernel with eBPF and Rust

ABOUT ME

- Michal Rostecki
- **vadorovsky** @ Github, Discord, Twitter etc.
- Software Engineer @ Deepfence Inc
- Rustacean with Go, C and Python background

WHAT IS EBPF?

- **extended Berkeley Packet Filter**
- A virtual machine with its own instruction set which runs sandboxed programs.
- Originated in the Linux kernel.

EVENT-DRIVEN

- Triggered by events in the kernel.
- Receiving pointers to kernel or userspace memory.

WHAT EVENTS?

NETWORK TRAFFIC

- **XDP** - ingress trafic on NIC driver
- **Classifier** - both directions, attached to qdisc

FUNCTION CALLS

- **Kprobes** - kernel functions
- **Uprobes** - userspace functions

SECURITY-RELATED ACTIONS

- LSM hooks

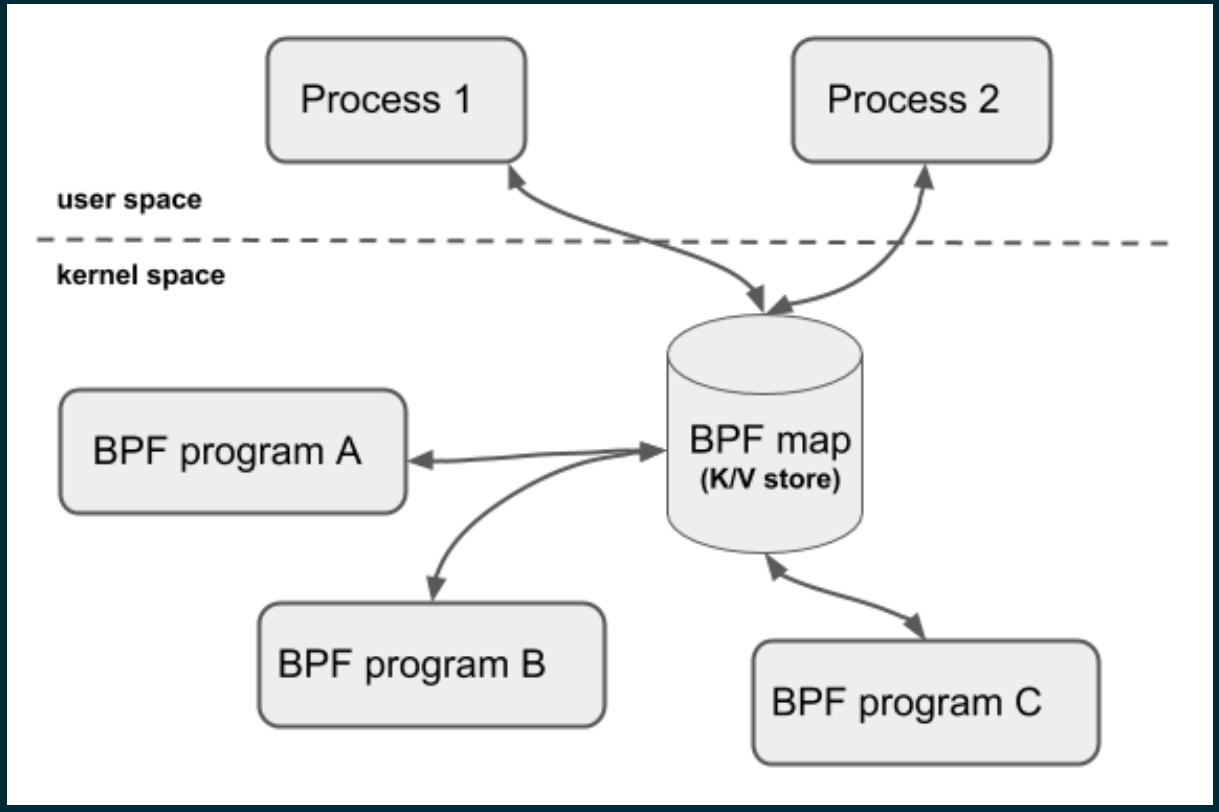
AND MANY MANY MORE...

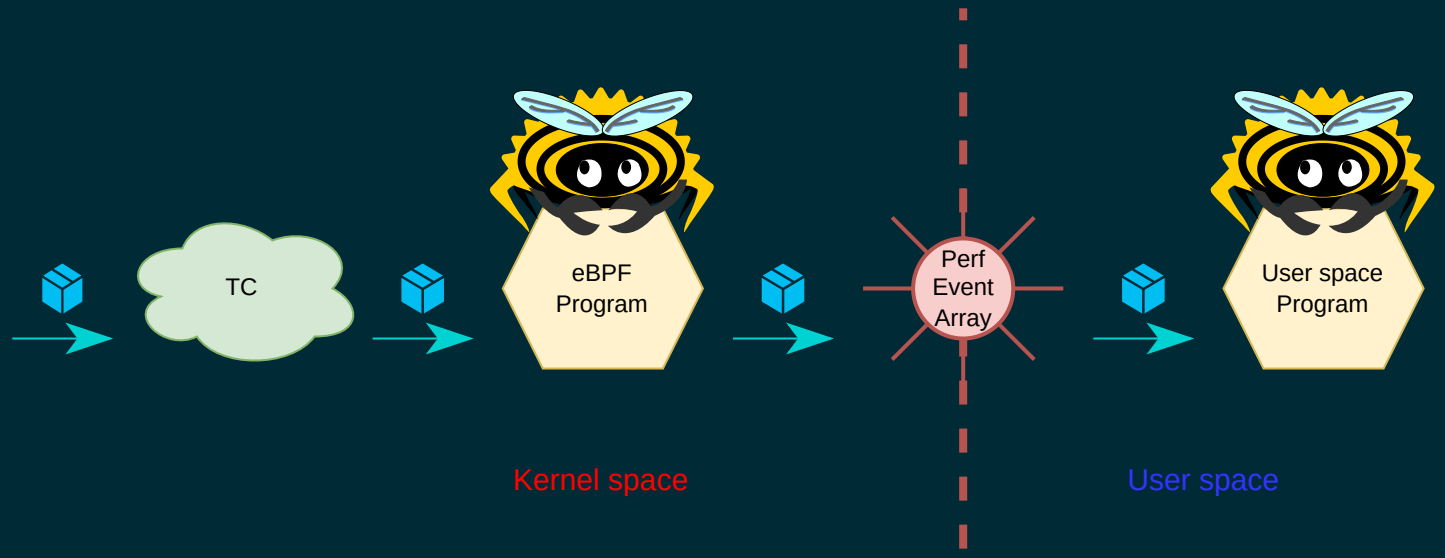
EBPF PROJECT LAYOUT

- **eBPF programs**: running in the kernel and reacting to events
- **userspace program**: loading eBPF programs to the kernel, interacting with them

BPF MAPS

- Storage for sharing data between eBPF programs (in kernel) and userspace.
- Different types:
  - hash maps
  - arrays
  - ring / perf buffers


Process 1
Process 2
user space
kernel space
BPF program A
BPF map
(K/V store)
BPF program B
BPF program C


TC
eBPF
Program
Perf
Event
Array
User space
Program
Kernel space
User space

EXAMPLES OF EBPF PROJECTS

- **Cilium** - Kubernetes networking, service mesh
- **Tracee** - application tracing tool
- **Falco** - security monitoring for containers

(those are in Go in C)

EXAMPLES OF EBPF PROJECTS USING AYA

- **bpfd** - daemon managing XDP programs for network filtering
- **Pulsar** - runtime observability tool for IoT

"OFFICIAL" WAYS TO USE EBPF

- libbpf
- libbpfgo
- libbpf-rs
- bcc

(With all of these, eBPF code is still in C)

AYA

Everything in Rust, even the eBPF programs!

WHY AYA?

RUST AND LINUX KERNEL

- There is Rust for Linux.
- There is an ongoing race to rewrite parts of the kernel in eBPF as well.
- **Aya says: why not both?**

MEMORY SAFETY?

- Actually... Rust memory model doesn't matter that much in eBPF part.
- eBPF has access to kernel and userspace pointers, which are unsafe.
- Memory safety relies more on the **eBPF verifier**.

C AND TYPE SAFETY

```
SEC("xdp")
int incorrect_xdp(struct __sk_buff *skb) {
    return XDP_PASS;
}
```

We made a mistake! We should have used

```
struct xdp_md *ctx
```

instead of

```
struct __sk_buff *skb
```

```
$ clang -O2 -emit-llvm -c incorrect_xdp.c -o - | llc -march=bpf -
$
```

But it compiles anyway, huh.

RUST AND TYPE SAFETY

```
#[xdp(name = "incorrect_xdp")]
pub fn incorrect_xdp(ctx: SkBuffContext) -> u32 {
    xdp_action::XDP_PASS
}
```

```
$ cargo xtask build-ebpf
[...]
error[E0308]: mismatched types
 --> src/main.rs:7:1
  |
7 | #[xdp(name = "incorrect_xdp")]
  | ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ expected struct `SkBuffContext
8 | pub fn incorrect_xdp(ctx: SkBuffContext) -> u32 {
[...]
```

It does not compile, nice.

ERROR HANDLING IN C

```
struct {
    __uint(type, BPF_MAP_TYPE_HASH);
    __uint(max_entries, 1024);
    __type(key, pid_t);
    __type(value, u32);
} pids SEC(".maps");
```

```
SEC("fentry/kernel_clone")
int BPF_PROG(kernel_clone, struct kernel_clone_args *args)
{
    /* Get the pid */
    pid_t pid = bpf_get_current_pid_tgid() >> 32;
    /* Save the pid in map */
    u32 val = 0;
    int err = bpf_map_update_elem(&pids, &pid, &val, 0);
    if (err < 0)
        return err;
    return 0;
}
```

ERROR HANDLING IN RUST

```
#[map(name = "pids")]
static mut PIDS: HashMap<u32, u32> = HashMap::<u32, u32>::with_ma

#[fentry(name = "kernel_clone")]
pub fn kernel_clone(ctx: FEntryContext) -> u32 {
    match try_kernel_clone(ctx) {
        Ok(ret) => ret,
        Err(_) => 1,
    }
}

fn try_kernel_clone(ctx: FEntryContext) -> Result<u32, c_long> {
    // Get the pid
    let pid = ctx.pid();
    // Save the pid in map.
    unsafe { PIDS.insert(&pid, &0, 0)? };
    Ok(0)
}
```

RUST TOOLCHAIN IS ALL YOU NEED

- No need for installing libbpf, make, clang or any C libraries in your system.

AYA-LOG

Logging library, based on perf buffers, production-ready.

```
#[fentry(name = "kernel_clone")]
pub fn kernel_clone(ctx: FEntryContext) -> u32 {
    let pid = ctx.pid();
    info!(&ctx, "new process: pid: {}", pid);
    0
}
```


vadorovsky ./aya-examples/clone main !? v1.60.0 14:44 sudo ./target/debug/clone
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:106] [FEAT PROBE] BPF program name support: true
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:109] [FEAT PROBE] BTF support: true
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:113] [FEAT PROBE] BTF func support: true
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:116] [FEAT PROBE] BTF global func support: true
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:122] [FEAT PROBE] BTF var and datasec support: true
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:128] [FEAT PROBE] BTF float support: true
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:131] [FEAT PROBE] BTF decl_tag support: true
12:44:52 [DEBUG] (1) aya::bpf: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/bpf.rs:134] [FEAT PROBE] BTF type_tag support: true
12:44:52 [DEBUG] (1) aya::obj::relocation: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/obj/relocation.rs:270] relocating program kernel_clone function kernel_clone
12:44:52 [DEBUG] (1) aya::obj::relocation: [/home/vadorovsky/.cargo/registry/src/github.com-1ecc6299db9ec823/aya-0.11.0/src/obj/relocation.rs:363] finished relocating program kernel_clone function kernel_clone
12:44:52 [INFO] clone: [src/main.rs:17] new process: pid: 183839
12:44:52 [INFO] clone: [src/main.rs:17] new process: pid: 187458
12:44:52 [INFO] clone: [src/main.rs:17] new process: pid: 187460
12:44:52 [INFO] clone: [src/main.rs:17] new process: pid: 183839
12:44:52 [INFO] clone: [clone/src/main.rs:46] Waiting for Ctrl-C...
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187313
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 183839
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187463
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 183839
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 22253
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 22253
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 22253
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187469
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187469
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187469
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187469
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187469
12:44:53 [INFO] clone: [src/main.rs:17] new process: pid: 187347

ASYNC SUPPORT

- Userspace part of projects can be asynchronous.
- Support for Tokio and async-std.
- aya-template uses Tokio by default.

HOW TO START WITH AYA?

```
rustup toolchain install nightly --component rust-src
cargo install bpf-linker
cargo install cargo-generate
```

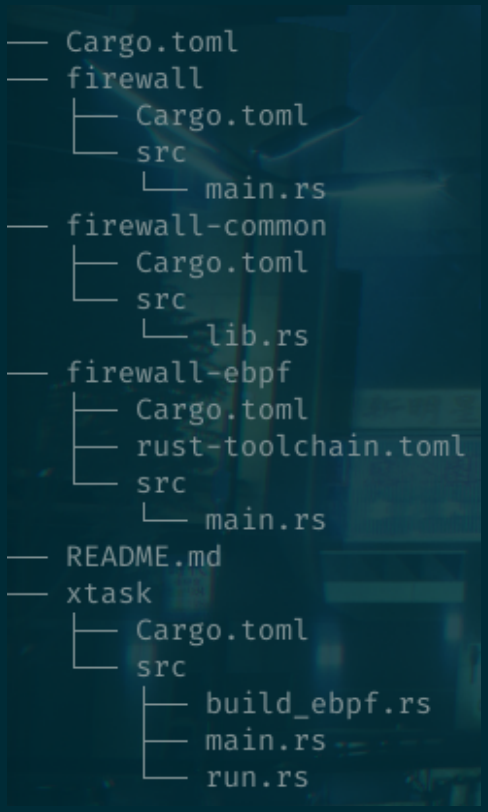
AYA-TEMPLATE

Creating a new project:

```
cargo install cargo-generate
cargo generate https://github.com/aya-rs/aya-template
```


vadorovsky ~/playground ♥ 14:05 ⟩ cargo generate https://github.com/aya-rs/aya-template
⚠ Unable to load config file: /home/vadorovsky/.cargo/cargo-generate.toml
⚠ Favorite https://github.com/aya-rs/aya-template not found in config, using it as a git repo url
Project Name : firewall
Generating template ...
? Which type of eBPF program?
kprobe
kretprobe
fentry
fexit
uprobe
uretprobe
sock_ops
socket_filter
sk_msg
xdp
❯ classifier
cgroup_skb
cgroup_sysctl
cgroup_sockopt
tracepoint
lsm
tp_btf

PROJECT LAYOUT



RUNNING THE PROJECT

```
cargo xtask run
```

Only building:

```
cargo xtask build-ebpf
cargo build
```

WHAT'S MISSING?

- Support for CO-RE in eBPF part of Aya
- That will require more work in Rust compiler

LET'S SEE A SIMPLE FIREWALL IN ACTION!

THANK YOU

- aya-rs.dev
- github.com/aya-rs/aya
- Discord