eBPF Deep Dive: Architecture, Implementation, and Real-world Applications

Cloud-Native & Platform Engineering Meetup

Здравейте

- Димитър Каналиев
- DevOps @ Siteground
- Харесвам OS технологии
- Най-вече eBPF

За какво ще си говорим?

- Защо съществува eBPF?
- Как работи?
- Какво можем да правим с eBPF?
- Какво правят други хора с него?

Проблемът

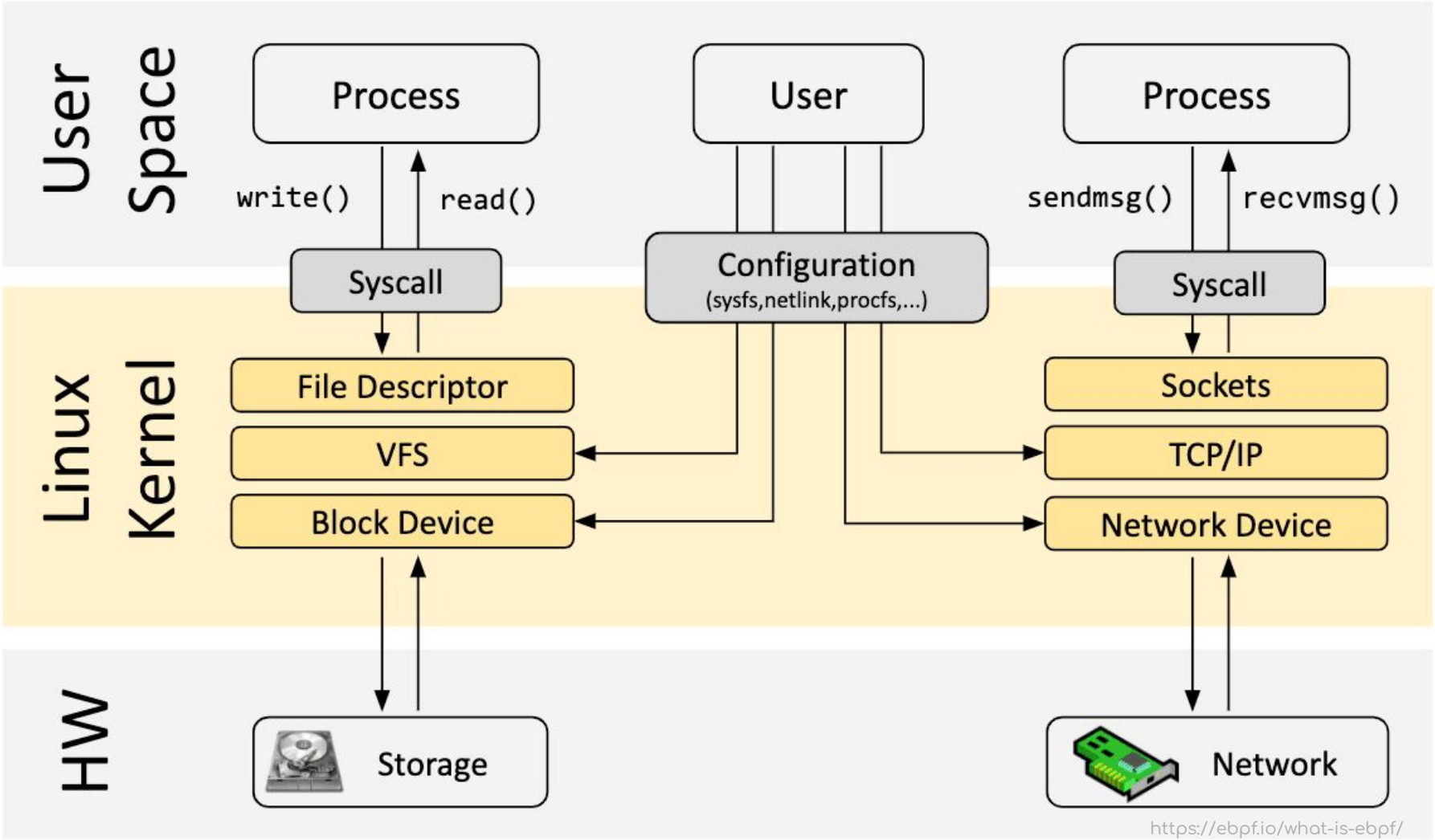
Mission critical

Логове

Метрики

Monitoring

Changelog


User Space
Process
User
Process
write()
read()
sendmsg()
recvmsg()
Syscall
Configuration
(sysfs,netlink,procfs,...)
Syscall
Linux Kernel
File Descriptor
Sockets
VFS
TCP/IP
Block Device
Network Device
HW
Storage
Network
https://ebpf.io/what-is-ebpf/

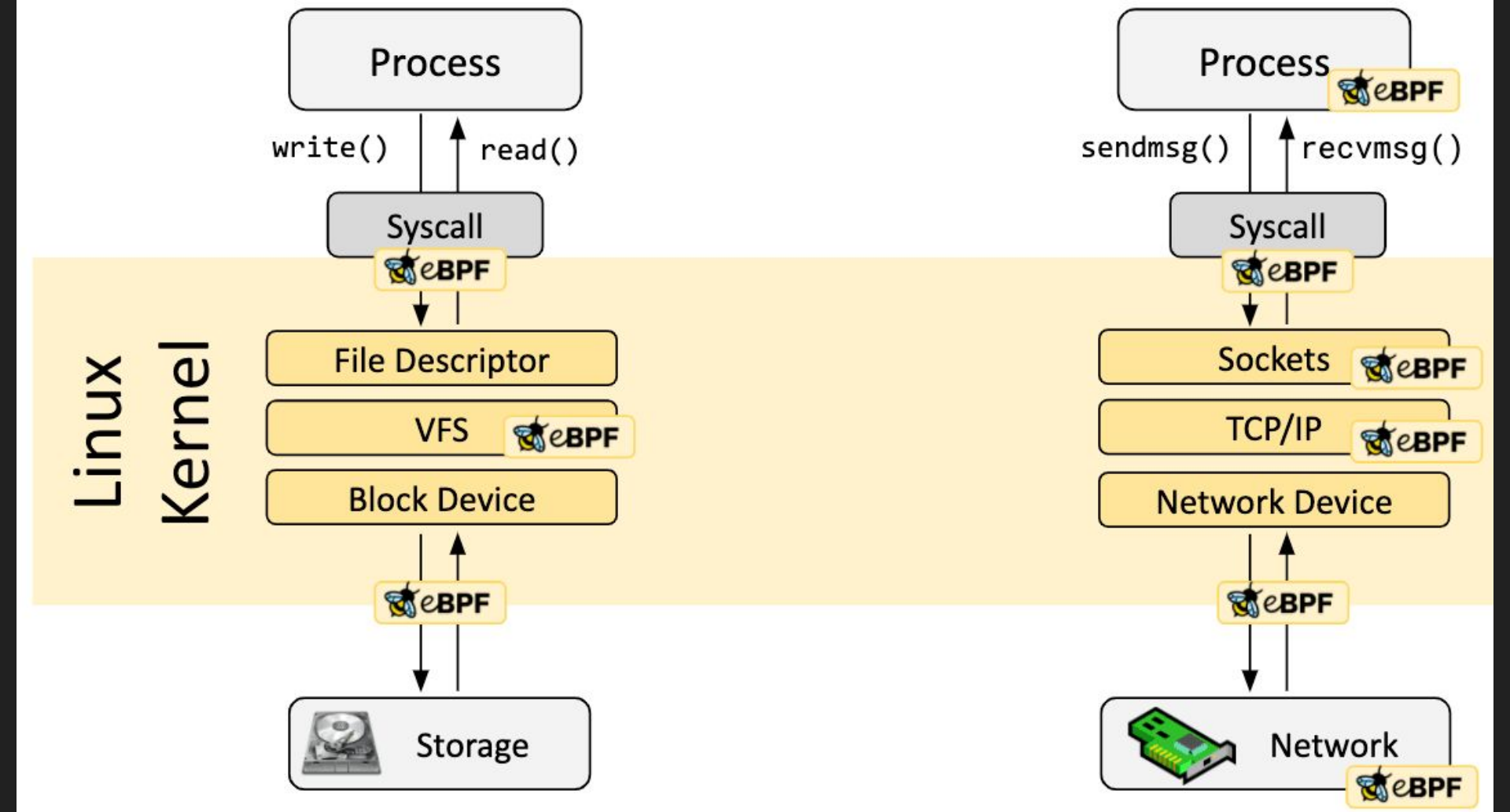
Програмируемост

Kmod

Live patches

Recompiling

eBPF


Process
write()
read()
Syscall
eBPF
File Descriptor
VFS
eBPF
Block Device
eBPF
Storage
Linux Kernel
Process
eBPF
sendmsg()
recvmsg()
Syscall
eBPF
Sockets
eBPF
TCP/IP
eBPF
Network Device
eBPF
Network
eBPF
https://ebpf.io/what-is-ebpf/

```c
#include <stdio.h>
#include <string.h>

void keep_secret(const char* secret) {
   printf("Not sharing the secret\n");
}

int main(int argc, char* argv[]) {
   if (argc != 2) {
       printf("Usage: %s <secret>\n", argv[0]);
       return 1;
   }

   keep_secret(argv[1]);
   return 0;
}
```

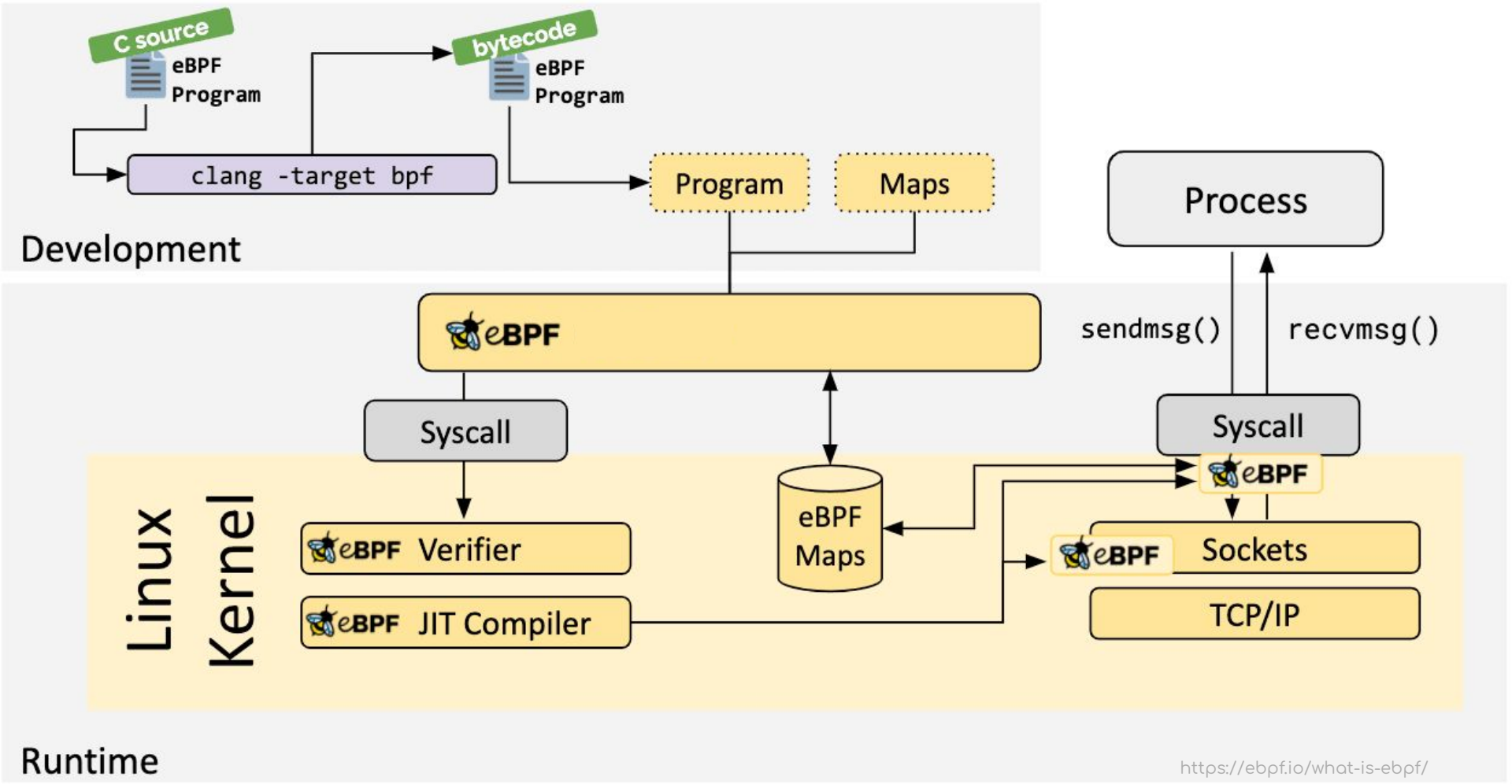
```
main:
        push    rbp
        mov     rbp, rsp
        mov     rdi, [rsi+8]    # argv[1]
        call    keep_secret
        xor     eax, eax        # return 0
        pop     rbp
        ret

keep_secret:
        push    rbp
        mov     rsi, rdi        # secret string
        mov     edi, .LC0
        xor     eax, eax
        call    printf
        pop     rbp
        ret
```

```
main:
        push    rbp
        mov     rbp, rsp
        mov     rdi, [rsi+8]    # argv[1]
        call    keep_secret
        xor     eax, eax        # return 0
        pop     rbp
        ret

keep_secret:
        push    rbp
        mov     rsi, rdi        # secret string
        mov     edi, .LC0
        xor     eax, eax
        call    printf
        pop     rbp
        ret
```

```c
#include <uapi/linux/ptrace.h>

int trace_keep_secret(struct pt_regs *ctx) {
    char secret[128];
    bpf_probe_read_user_str(secret, sizeof(secret),
    (void*)PT_REGS_PARM1(ctx));
    bpf_trace_printk("Secret captured: %s\\n", secret);
    return 0;
}
```

```
#!/usr/bin/python3
from bcc import BPF

prog = """
    # eBPF program goes here
"""

b = BPF(text=prog)

b.attach_uprobe(name="program", sym="keep_secret",
fn_name="trace_keep_secret")

while True:
   try:
       (task, pid, cpu, flags, ts, msg) = b.trace_fields()
       print(f"[{pid}] {msg.decode()}")
   except KeyboardInterrupt:
       break
```

```
Disassembly of section uprobe/print_secret:

0000000000000000 <trace_keep_secret>:
       0:    bf 13 00 00 00 00 00 00 r3 = r1
       1:    bf a6 00 00 00 00 00 00 r6 = r10
       2:    07 06 00 00 80 ff ff ff r6 += -0x80
       3:    bf 61 00 00 00 00 00 00 r1 = r6
       4:    b7 02 00 00 80 00 00 00 r2 = 0x80
       5:    85 00 00 00 72 00 00 00 call 0x72
       6:    18 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 r1 = 0x0
ll
       8:    b7 02 00 00 15 00 00 00 r2 = 0x15
       9:    bf 63 00 00 00 00 00 00 r3 = r6
      10:    85 00 00 00 06 00 00 00 call 0x6
      11:    b7 00 00 00 00 00 00 00 r0 = 0x0
      12:    95 00 00 00 00 00 00 00 exit
```


C source
eBPF
Program
bytecode
eBPF
Program
clang -target bpf
Program
Maps
Development
Process
eBPF
sendmsg()
recvmsg()
Syscall
Syscall
eBPF
eBPF Verifier
eBPF
Maps
eBPF
eBPF Sockets
eBPF JIT Compiler
TCP/IP
Linux
Kernel
Runtime
https://ebpf.io/what-is-ebpf/

Проблемът

```
$ sudo syscount -p $(pidof fintech-service)

SYSCALL            COUNT
getrandom          7878397
read               2382711
write              1674411
```

Проблемът

```
$ sudo syscount -L -p $(pidof fintech-service)

SYSCALL                      COUNT        TIME (us)
getrandom                        4      3415860.022
nanosleep                      291        12038.707
ftruncate                        1          122.939
write                            4           63.389
```

Проблемът

```
# cat /proc/sys/kernel/random/entropy_avail
8
```

BCC

tools/argdist: Display function parameter values as a histogram or frequency count.

tools/deadlock: Detect potential deadlocks on a running process.

tools/uobjnew: Summarize object allocation events by object type and number of bytes allocated.

tools/memleak: Display outstanding memory allocations to find memory leaks

И още 100+ подобни

bpftrace

```
$ sudo bpftrace -e '
    uprobe:./secret_printer:print_secret {
        printf("Secret captured: %s\n", str(arg0));
    }
    '
```

bpftrace

Files opened, for processes in the root cgroup-v2

```
$ bpftrace -e 'tracepoint:syscalls:sys_enter_openat /cgroup ==
cgroupid("/sys/fs/cgroup/unified/mycg")/ { printf("%s\n",
str(args->filename)); }'
```

Files opened by process

```
$ bpftrace -e 'tracepoint:syscalls:sys_enter_open { printf("%s %s\n",
comm, str(args->filename)); }'
```

Syscall count by program

```
$ bpftrace -e 'tracepoint:raw_syscalls:sys_enter { @[comm] = count();
}'
```

libbpf

```c
#include "vmlinux.h"
#include <bpf/bpf_helpers.h>

SEC("uprobe/print_secret")
int trace_print_secret(const char *secret)
{
    char str[128];
    bpf_probe_read_user_str(str, sizeof(str), secret);
    bpf_printk("Secret captured: %s\n", str);
    return 0;
}

char LICENSE[] SEC("license") = "GPL";
```

ebpf-go

```
package main

//go:generate go run github.com/cilium/ebpf/cmd/bpf2go counter
counter.c

func loadBpfObjects(obj interface{}, opts *ebpf.CollectionOptions)
error {
    spec, err := loadBpf()
    if err != nil {
        return err
    }

    return spec.LoadAndAssign(obj, opts)
}
```

Aya

```
#[xdp]
pub fn xdp_hello(ctx: XdpContext) -> u32 {
    match unsafe { try_xdp_hello(ctx) } {
        Ok(ret) => ret,
        Err(_) => xdp_action::XDP_ABORTED,
    }
}

unsafe fn try_xdp_hello(ctx: XdpContext) -> Result<u32, u32> {
    info!(&ctx, "received a packet");
    Ok(xdp_action::XDP_PASS)
}
```

rbpf

```
extern crate rbpf;

fn main() {
    let prog = &[
        0xb4, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,// mov32 r0, 0
        0xb4, 0x01, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,// mov32 r1, 2
        0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,// add32 r0, 1
        0x0c, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,// add32 r0,r1
        0x95, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 // exit
    ];

    let vm = rbpf::EbpfVmNoData::new(Some(prog)).unwrap();

    assert_eq!(vm.execute_program().unwrap(), 0x3);
}
```

hello-ebpf

```
@BPF(license = "GPL")

public abstract class HelloWorld extends BPFProgram implements
SystemCallHooks {

    @Override
    public void enterOpenat2(int dfd, String filename, Ptr<open_how>
how) {
        bpf_trace_printk("Hello, World!");
    }

    public static void main(String[] args) {
        try (HelloWorld program = BPFProgram.load(HelloWorld.class)){
        ...
```

Hooks

BPF_PROG_TYPE_KPROBE

BPF_PROG_TYPE_SCHED_CLS

BPF_PROG_TYPE_XDP

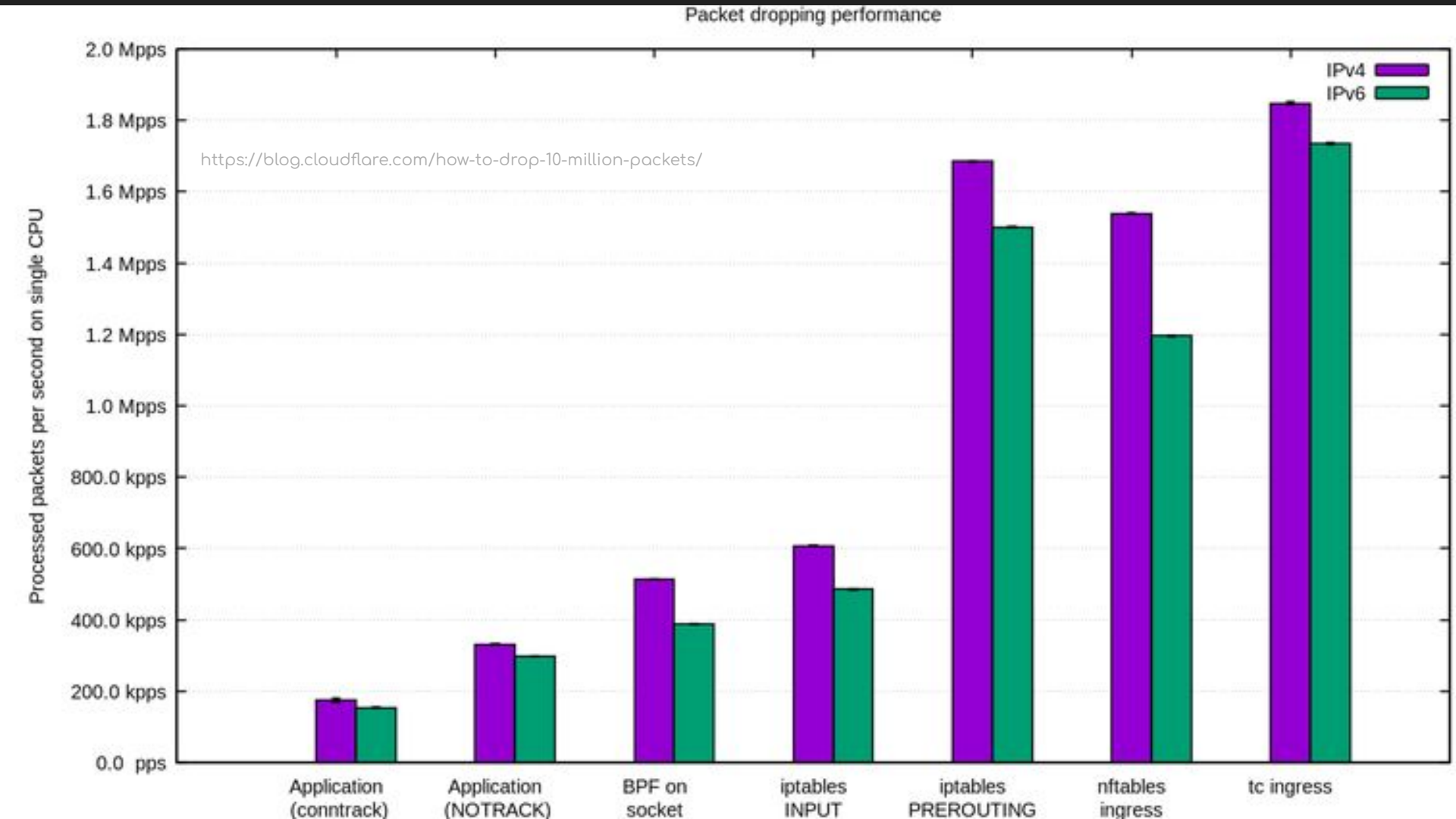
BPF_PROG_TYPE_LSM

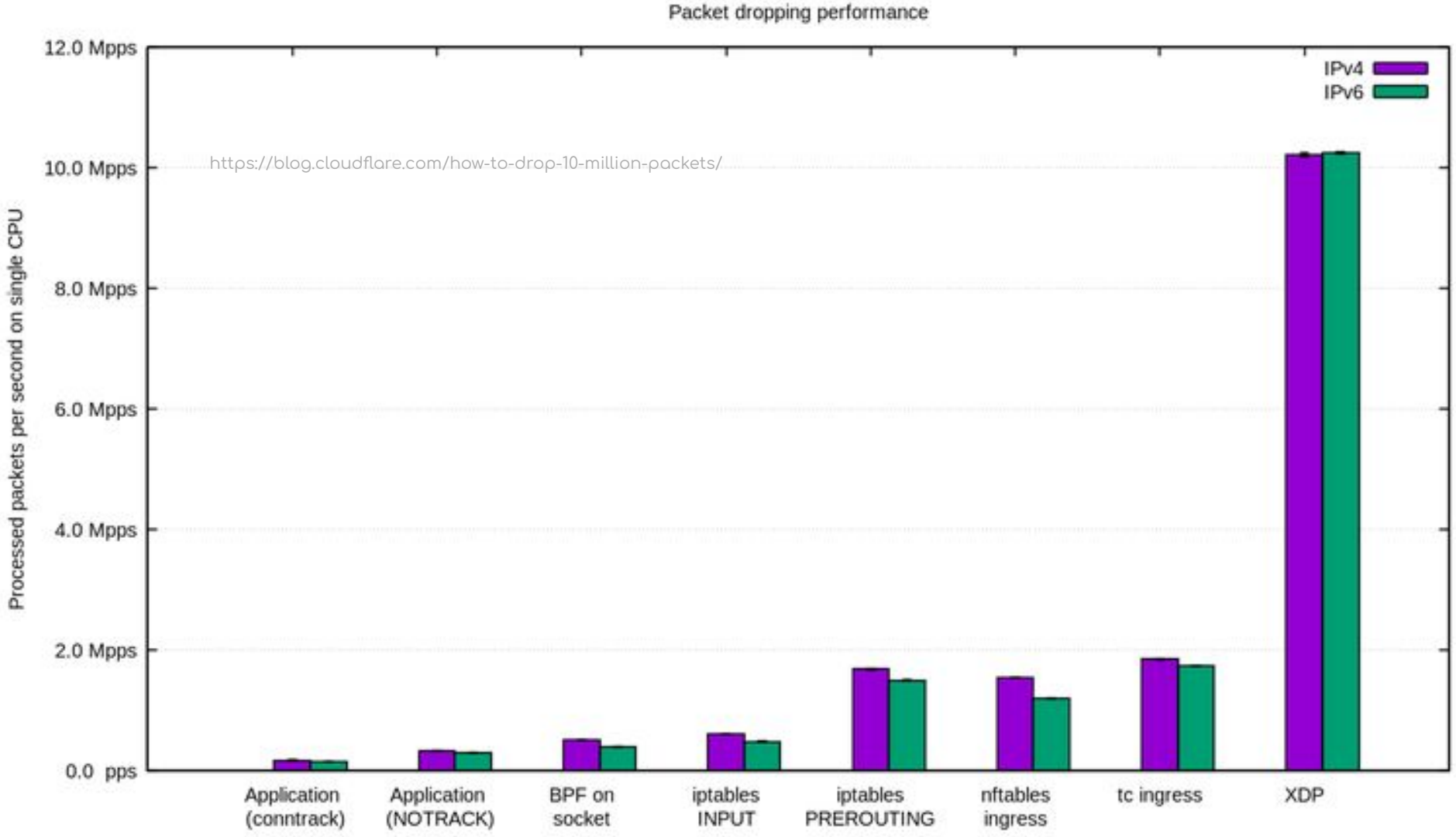
BPF_PROG_TYPE_CGROUP_*

И много други

TC / XDP

```
SEC("xdp")
int drop_all(struct xdp_md* ctx) {
    void* data = (void*)(long)ctx->data;
    void* data_end = (void*)(long)ctx->data_end;

    return XDP_DROP;
}

char __license[] SEC("license") = "GPL";
```


Packet dropping performance
https://blog.cloudflare.com/how-to-drop-10-million-packets/
Processed packets per second on single CPU
2.0 Mpps
1.8 Mpps
1.6 Mpps
1.4 Mpps
1.2 Mpps
1.0 Mpps
800.0 kpps
600.0 kpps
400.0 kpps
200.0 kpps
0.0 pps
IPv4
IPv6
Application (conntrack)
Application (NOTRACK)
BPF on socket
iptables INPUT
iptables PREROUTING
nftables ingress
tc ingress


Packet dropping performance
https://blog.cloudflare.com/how-to-drop-10-million-packets/
Processed packets per second on single CPU
12.0 Mpps
10.0 Mpps
8.0 Mpps
6.0 Mpps
4.0 Mpps
2.0 Mpps
0.0 pps
IPv4
IPv6
Application (conntrack)
Application (NOTRACK)
BPF on socket
iptables INPUT
iptables PREROUTING
nftables ingress
tc ingress
XDP

LSM

```
const __u32 blockme = 16843009; // 1.1.1.1

SEC("lsm/socket_connect")
int BPF_PROG(restrict_connect, struct socket *sock, struct sockaddr
*address, int addrlen, int ret)
{
    __u32 dest = addr->sin_addr.s_addr;
    bpf_printk("lsm: found connect to %d", dest);

    if (dest == blockme)
    {
        bpf_printk("lsm: blocking %d", dest);
        return -EPERM;
    }
    return 0;
}
```

CGROUP_*

```
SEC("cgroup_skb/egress")
int count_egress_packets(struct __sk_buff *skb) {
    u32 key      = 0;
    u64 init_val = 1;

    u64 *count = bpf_map_lookup_elem(&pkt_count, &key);
    if (!count) {
        bpf_map_update_elem(&pkt_count, &key, &init_val, BPF_ANY);
        return 1;
    }
    __sync_fetch_and_add(count, 1);

    return 1;
}
```

Cilium / Hubble / Tetragon

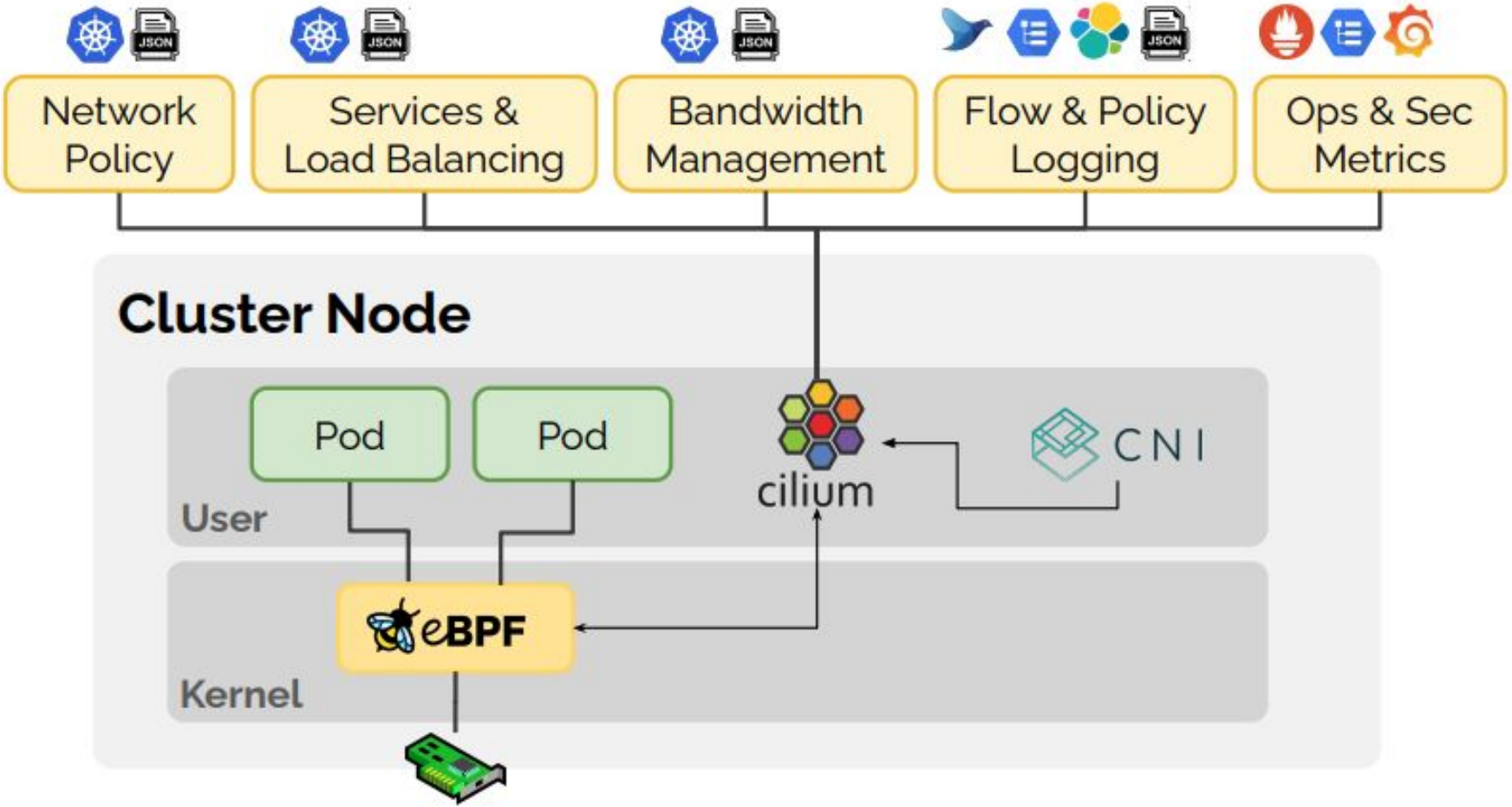
Performant container CNI

Fast networking

Process execution events

Communication visibility

Wide observability



https://ebpf.io/what-is-ebpf/

Falco

Real time security monitoring and cross-system alerts

pwru

0xffff888c43ea4ee8 1 ~bin/sshd:656815 4026531840 0 ens4:2 0x0800 1460 602 10.186.0.42:22->85.196.187.226:55088(tcp) skb_release_head_state

0xffff888c6f875c00 1 <empty>:0 4026531840 0 ens4:2 0x0800 1460 52 85.196.187.226:55088->10.186.0.42:22(tcp) tcp_v4_early_demux

0xffff888c6f875c00 1 <empty>:0 4026531840 0 ens4:2 0x0800 65536 52 85.196.187.226:55088->10.186.0.42:22(tcp) ip_local_deliver

0xffff888c53bf78e8 1 <empty>:0 4026531840 0 ens4:2 0x0800 1460 262 10.186.0.42:22->85.196.187.226:55088(tcp) validate_xmit_skb

0xffff888c53bf78e8 1 <empty>:0 4026531840 0 ens4:2 0x0800 1460 262 10.186.0.42:22->85.196.187.226:55088(tcp) netif_skb_features

sched_ext

Programmable Linux Kernel Scheduler

Beyla

Auto Instrumentation of RED metrics

Community

slack.cilium.io

reddit.com/r/ebpf

lore.kernel.org/bpf

ebpf.io/fosdem-2025

Getting started

https://ebpf.io/what-is-ebpf/

https://cilium.isovalent.com/hubfs/Learning-eBPF%20-%20Full%20book.pdf

https://github.com/xdp-project/xdp-tutorial

https://eunomia.dev/en/tutorials/1-helloworld/

https://cilium.io/labs/

Links

https://github.com/iovisor/bcc

https://github.com/bpftrace/bpftrace

https://github.com/aya-rs/aya

https://github.com/parttimenerd/hello-ebpf

https://github.com/libbpf

https://github.com/cilium/cilium

https://github.com/cilium/pwru

https://github.com/cilium/tetragon

https://github.com/cilium/hubble

https://github.com/falcosecurity/falco

https://github.com/grafana/beyla

https://github.com/libbpf/bpftool

Q&A