# Make your own firewall

...using eBPFs

August 19, 2020 ANNOUNCEMENTS

## Google announces Cilium & eBPF as the new networking dataplane for GKE

MONITORING

## eBPF and XDP for Processing Packets at Bare-metal Speed

Nedim Šabić on June 3, 2019

DEVELOPMENT / CONTRIBUTED

## How io_uring and eBPF Will Revolutionize Programming in Linux

21 Apr 2020 8:49am, by Glauber Costa

November 10, 2020 DEEP DIVE

## eBPF - The Future of Networking & Security

# ...but what is it?

- extended Berkeley Packet Filter
- A way to program the Linux kernel without crashing it
- Trace connections, program executions, file operations etc.
- You can also edit / block / redirect incoming packets
- Great tooling available (C, Python, Go, Rust)

# Demo time!

 nikofil/ebpf-firewall

# Useful tools: track file opens, Python function calls, even read decrypted TLS!

```
[root@c8nikos ~]# opensnoop
PID    COMM               FD ERR PATH
38793  tmux: server       11   0 /proc/38849/cmdline
38793  tmux: server       11   0 /proc/38849/cmdline
38793  tmux: server       11   0 /proc/38849/cmdline
38793  tmux: server       11   0 /proc/38849/cmdline
38793  tmux: server       11   0 /proc/38816/cmdline
38793  tmux: server       11   0 /proc/38849/cmdline
38851  cat                 3   0 /etc/ld.so.cache
38851  cat                 3   0 /lib64/libc.so.6
38851  cat                -1   2 /usr/lib/locale/locale-archive
38851  cat                 3   0 /usr/share/locale/locale.alias
38851  cat                 3   0 /usr/lib/locale/en_GB.utf8/LC_IDENTIFICATION
```

```
[root@c8nikos ~]# sslsniff
FUNC         TIME(s)            COMM             PID    LEN
WRITE/SEND   0.000000000        curl             38910  71
----- DATA -----
GET / HTTP/1.1
Host: cern.ch
User-Agent: curl/7.61.1
Accept: */*
```

# Thank you! Questions?

```
/* Map for blocking IP addresses from userspace */
struct bpf_map_def __section("maps") blocked_map = {
    .type = BPF_MAP_TYPE_HASH,
    .key_size = sizeof(__u32),
    .value_size = sizeof(__u32),
    .max_entries = 10000,
};

/* Handle a packet: send its information to userspace and return whether it should be allowed */
inline bool handle_pkt(struct __sk_buff *skb, bool egress) {
    struct iphdr iph;
    /* Load packet header */
    bpf_skb_load_bytes(skb, 0, &iph, sizeof(struct iphdr));
    /* Check if IPs are in "blocked" map */
    bool blocked = bpf_map_lookup_elem(&blocked_map, &iph.saddr) || bpf_map_lookup_elem(&blocked_map, &iph.daddr);
    if (iph.version == 4) {
        conn c = {
            .flags = egress | (blocked << 1),
            .srcip = iph.saddr,
            .dstip = iph.daddr,
        };

        /* Send packet info to user program to display */
        bpf_map_push_elem(&flows_map, &c, 0);
    }
    /* Return whether it should be allowed or dropped */
    return !blocked;
}
```