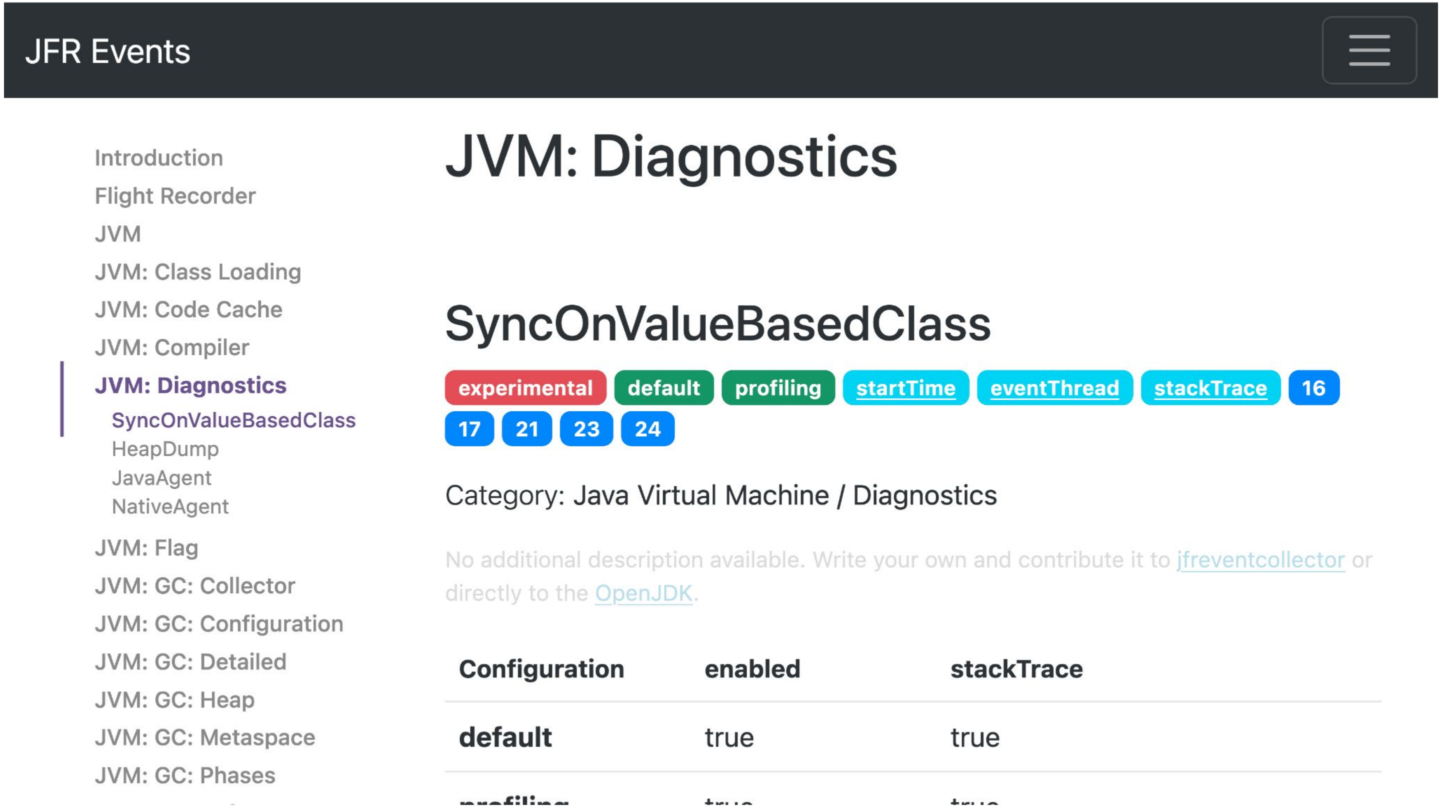
Building a Lightning Fast Firewall in Java & eBPF

Johannes Bechberger
mostlynerdless.de
OpenJDK Developer, SAP
Creator of hello-ebpf

Mohammed Aboullaite
@laytoun
Sr Backend Engineer, Spotify
Java Champion
Google Developer Expert

We have a simple web application



JFR Events

- Introduction
- Flight Recorder
- JVM
- JVM: Class Loading
- JVM: Code Cache
- JVM: Compiler
- JVM: Diagnostics
  - SyncOnValueBasedClass
  - HeapDump
  - JavaAgent
  - NativeAgent
- JVM: Flag
- JVM: GC: Collector
- JVM: GC: Configuration
- JVM: GC: Detailed
- JVM: GC: Heap
- JVM: GC: Metaspace
- JVM: GC: Phases

JVM: Diagnostics

SyncOnValueBasedClass

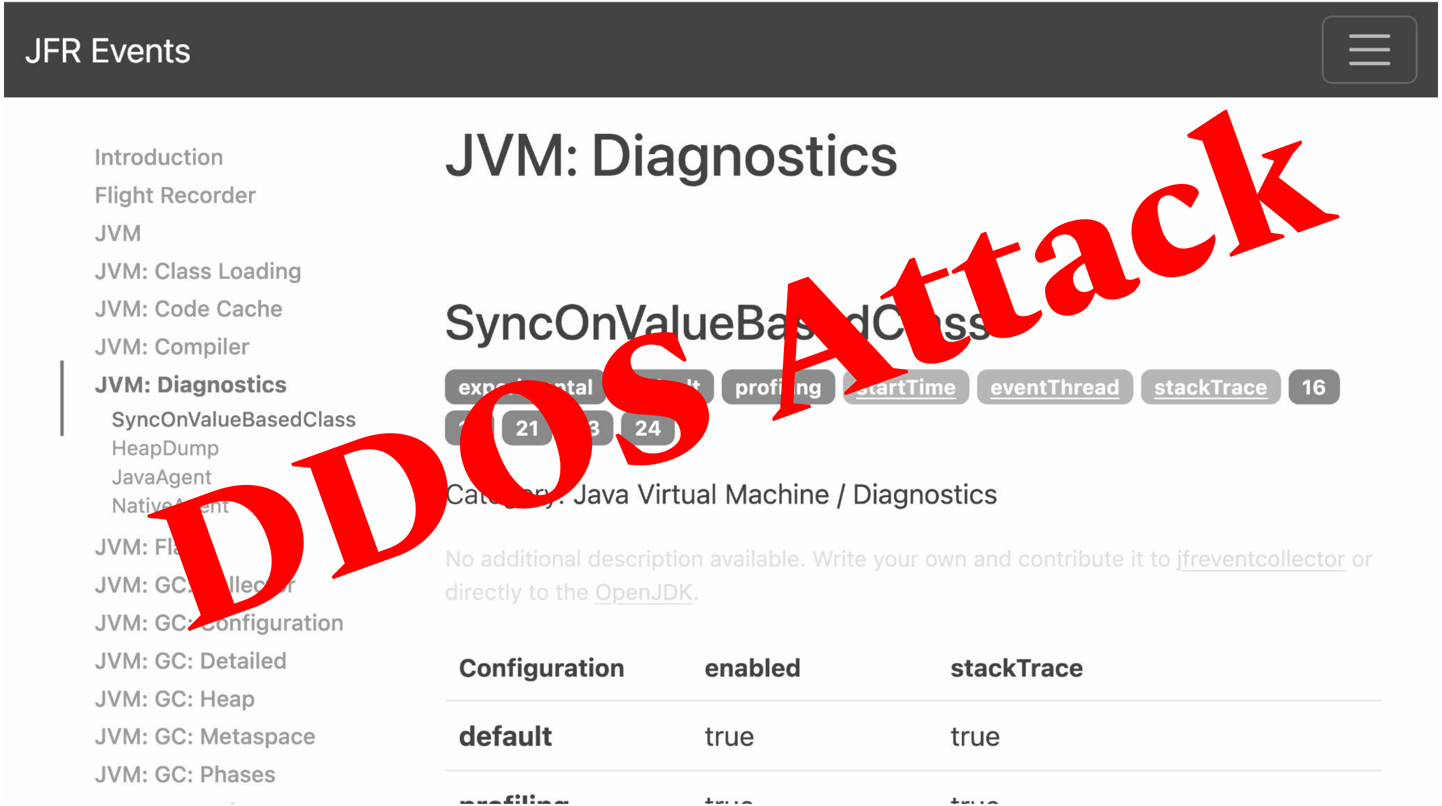
experimental default profiling startTime eventThread stackTrace 16 17 21 23 24

Category: Java Virtual Machine / Diagnostics

No additional description available. Write your own and contribute it to jfreventcollector or directly to the OpenJDK.

Configuration	enabled	stackTrace
default	true	true
profiling	true	true

We have a simple web application



JFR Events

- Introduction
- Flight Recorder
- JVM
- JVM: Class Loading
- JVM: Code Cache
- JVM: Compiler
- JVM: Diagnostics
  - SyncOnValueBasedClass
  - HeapDump
  - JavaAgent
  - NativeAgent
- JVM: Flag
- JVM: GC: Collector
- JVM: GC: Configuration
- JVM: GC: Detailed
- JVM: GC: Heap
- JVM: GC: Metaspace
- JVM: GC: Phases

JVM: Diagnostics

SyncOnValueBasedClass

experimental default profiling startTime eventThread stackTrace 16 17 21 23 24

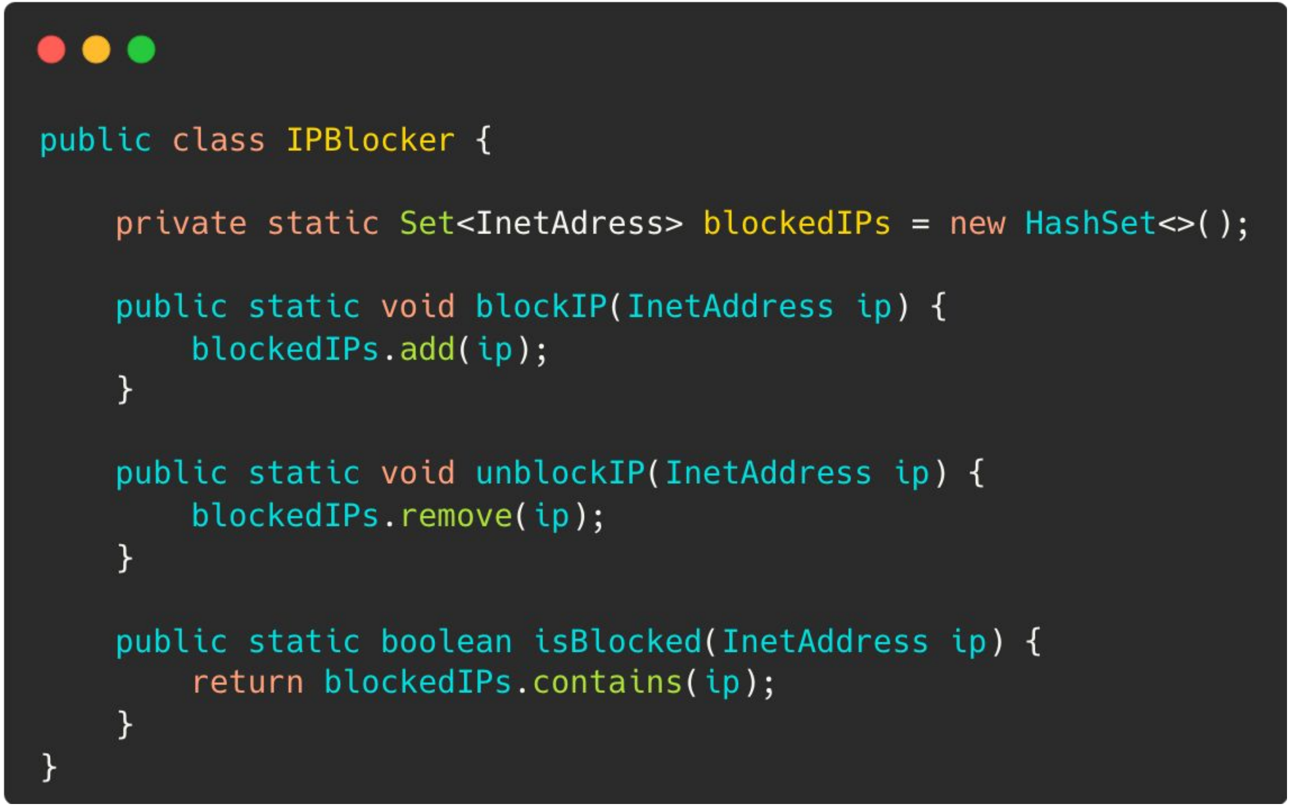
Category: Java Virtual Machine / Diagnostics

No additional description available. Write your own and contribute it to jfreventcollector or directly to the OpenJDK.

Configuration	enabled	stackTrace
default	true	true
profiling	true	true

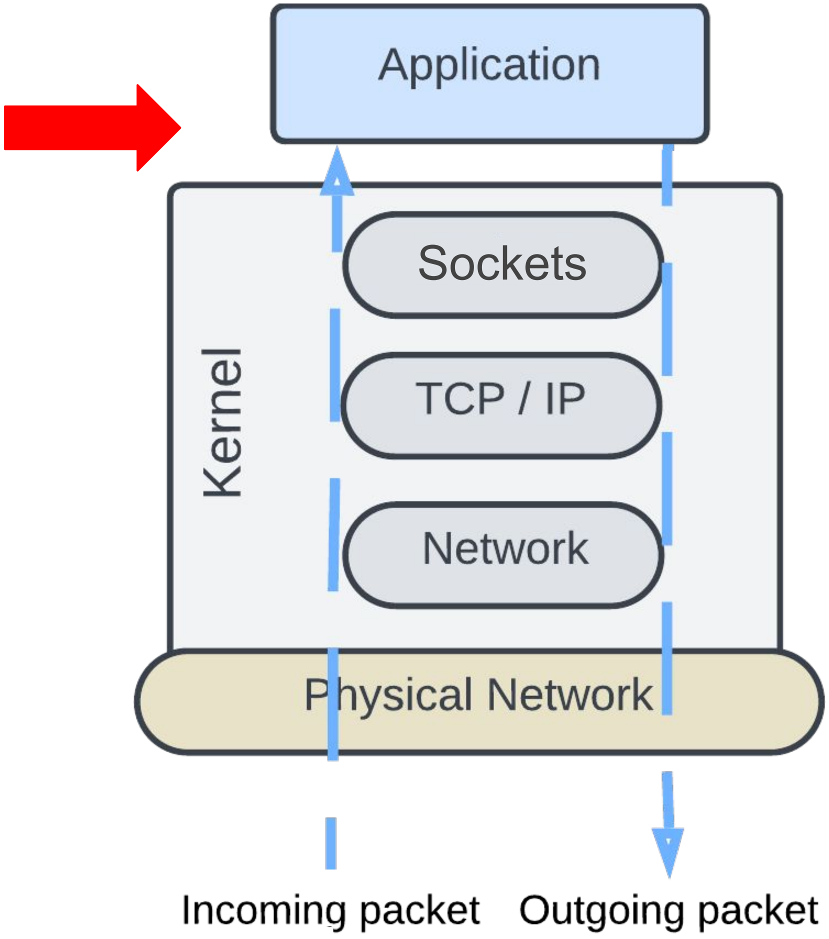
The naïve way?

Add it to your application

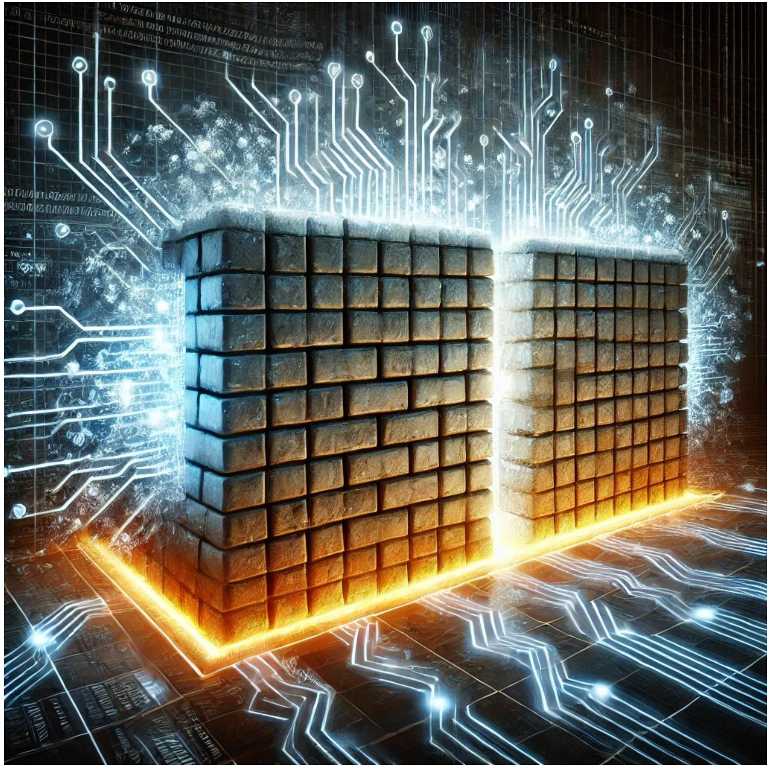


```java
public class IPBlocker {

    private static Set<InetAdress> blockedIPs = new HashSet<>();

    public static void blockIP(InetAddress ip) {
        blockedIPs.add(ip);
    }

    public static void unblockIP(InetAddress ip) {
        blockedIPs.remove(ip);
    }

    public static boolean isBlocked(InetAddress ip) {
        return blockedIPs.contains(ip);
    }
}
```

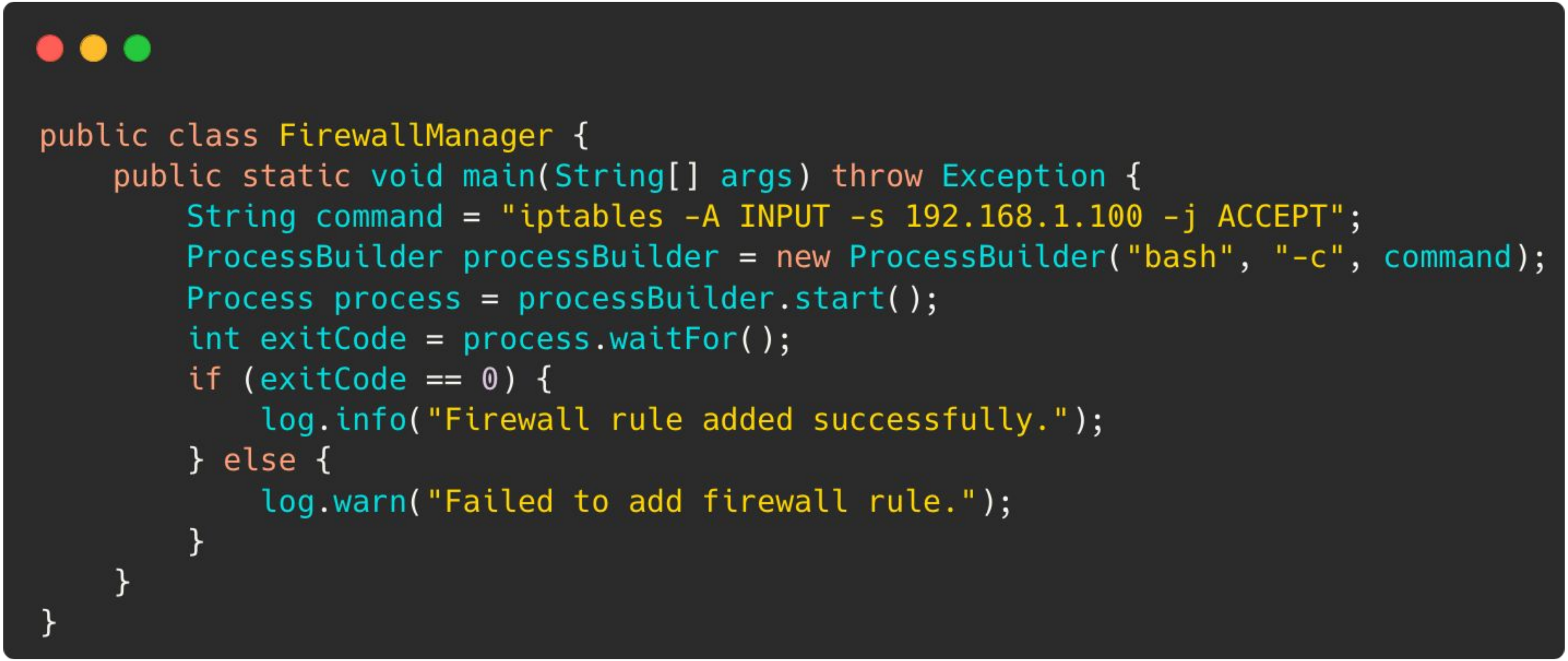
Any Problems?


Application
Kernel
Sockets
TCP / IP
Network
Physical Network
Incoming packet
Outgoing packet

Alternative: Use a Firewall

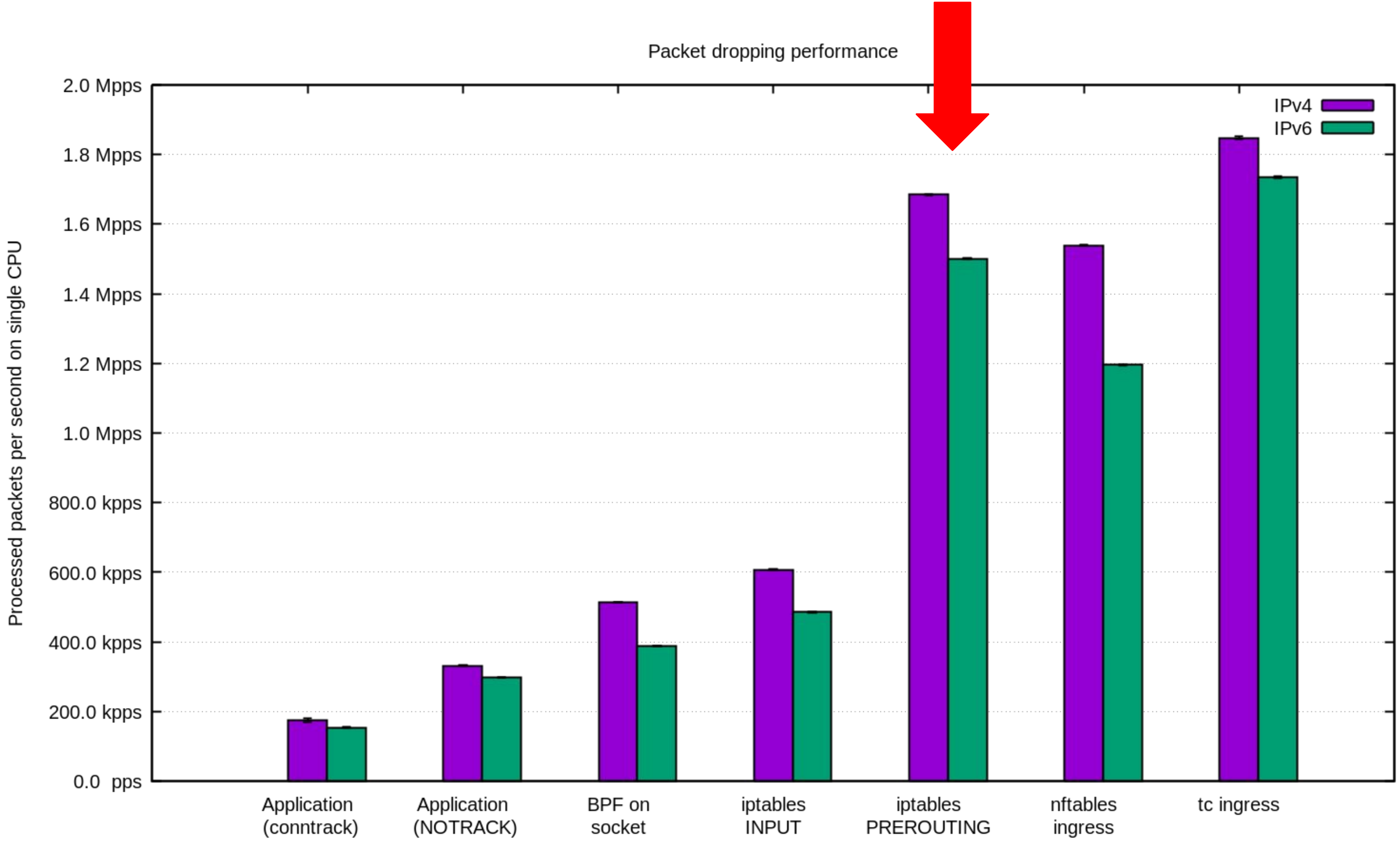


Naive implementation



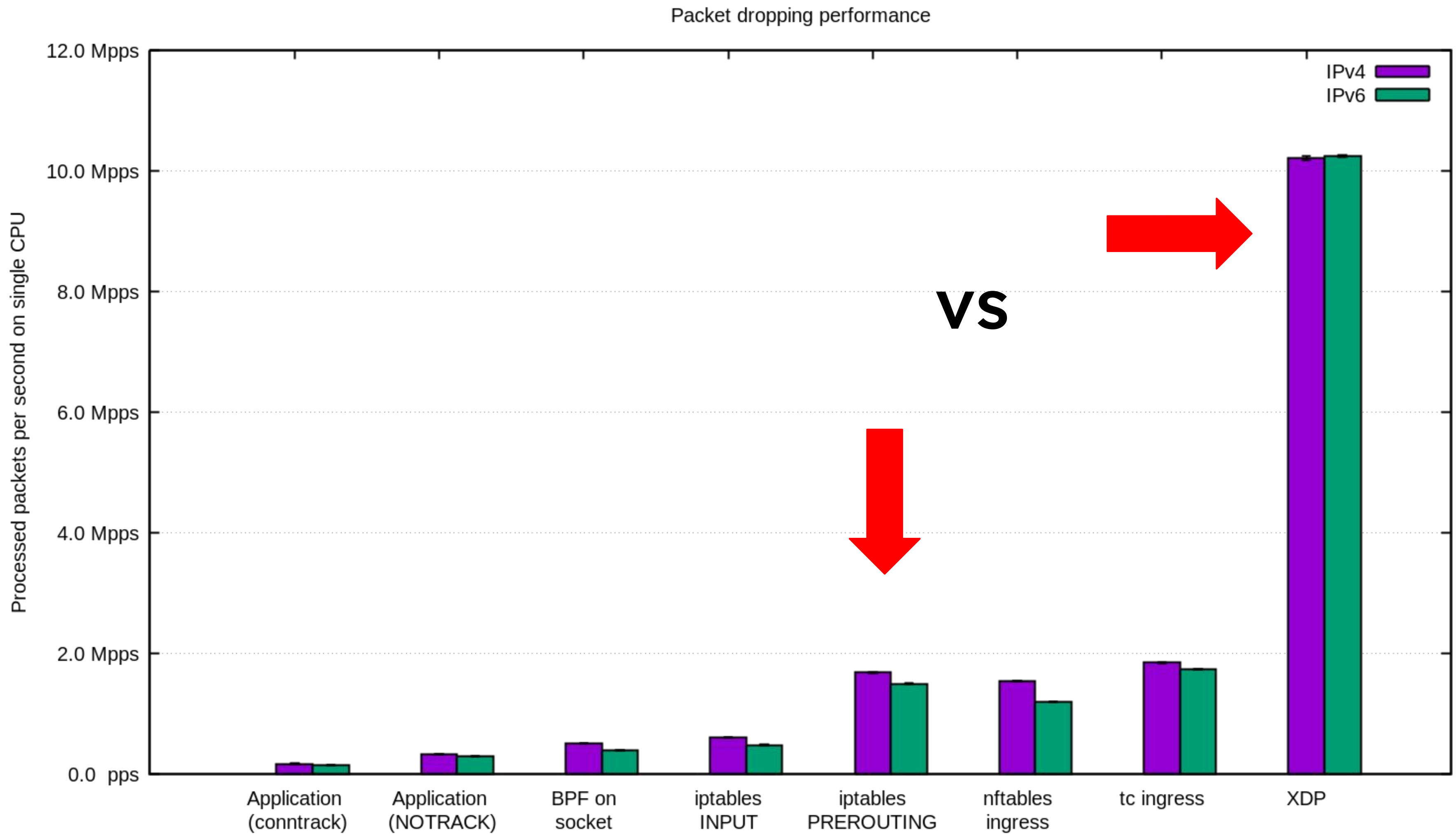
```
public class FirewallManager {
    public static void main(String[] args) throw Exception {
        String command = "iptables -A INPUT -s 192.168.1.100 -j ACCEPT";
        ProcessBuilder processBuilder = new ProcessBuilder("bash", "-c", command);
        Process process = processBuilder.start();
        int exitCode = process.waitFor();
        if (exitCode == 0) {
            log.info("Firewall rule added successfully.");
        } else {
            log.warn("Failed to add firewall rule.");
        }
    }
}
```

How to add more logic and improve speed?



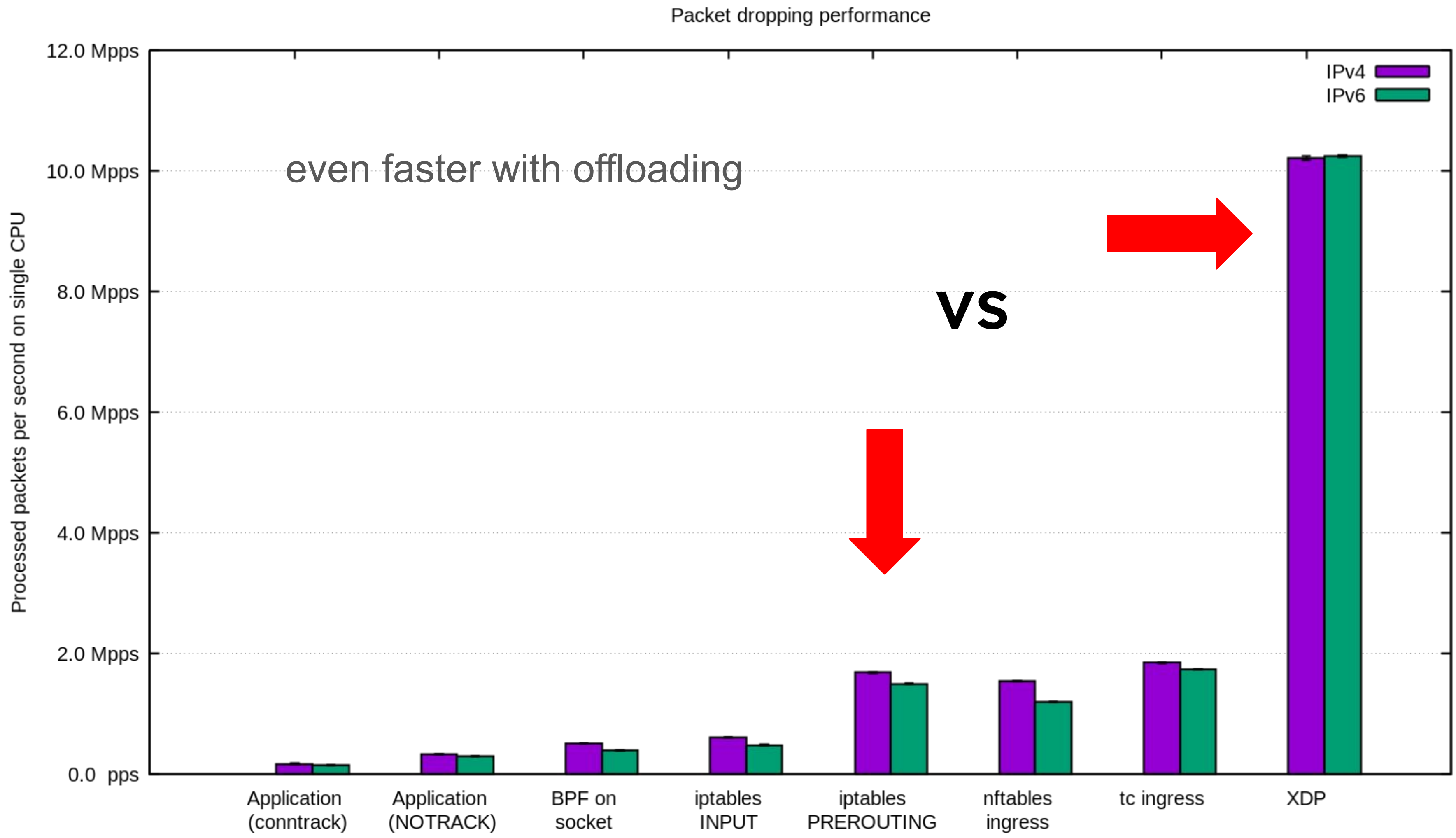
Source: https://blog.cloudflare.com/how-to-drop-10-million-packets/

Become a 10x Firewall



VS

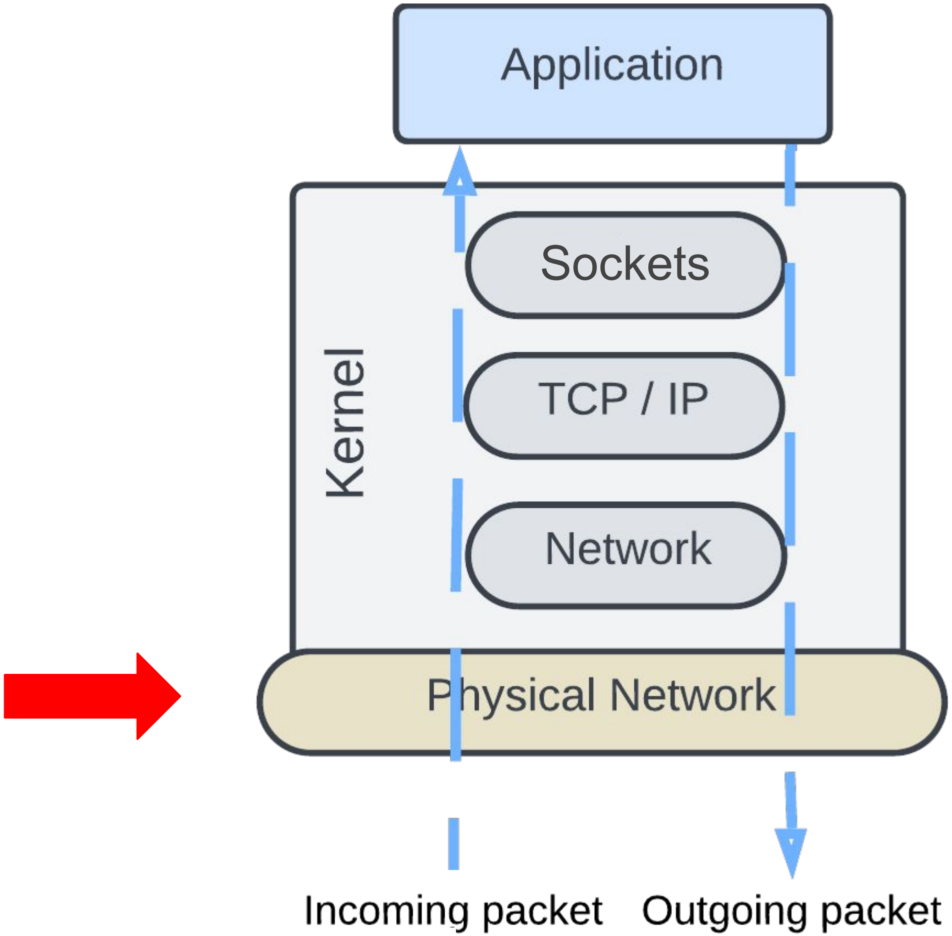
Source: https://blog.cloudflare.com/how-to-drop-10-million-packets/



even faster with offloading

VS

Source: https://blog.cloudflare.com/how-to-drop-10-million-packets/



Application

Kernel

Sockets

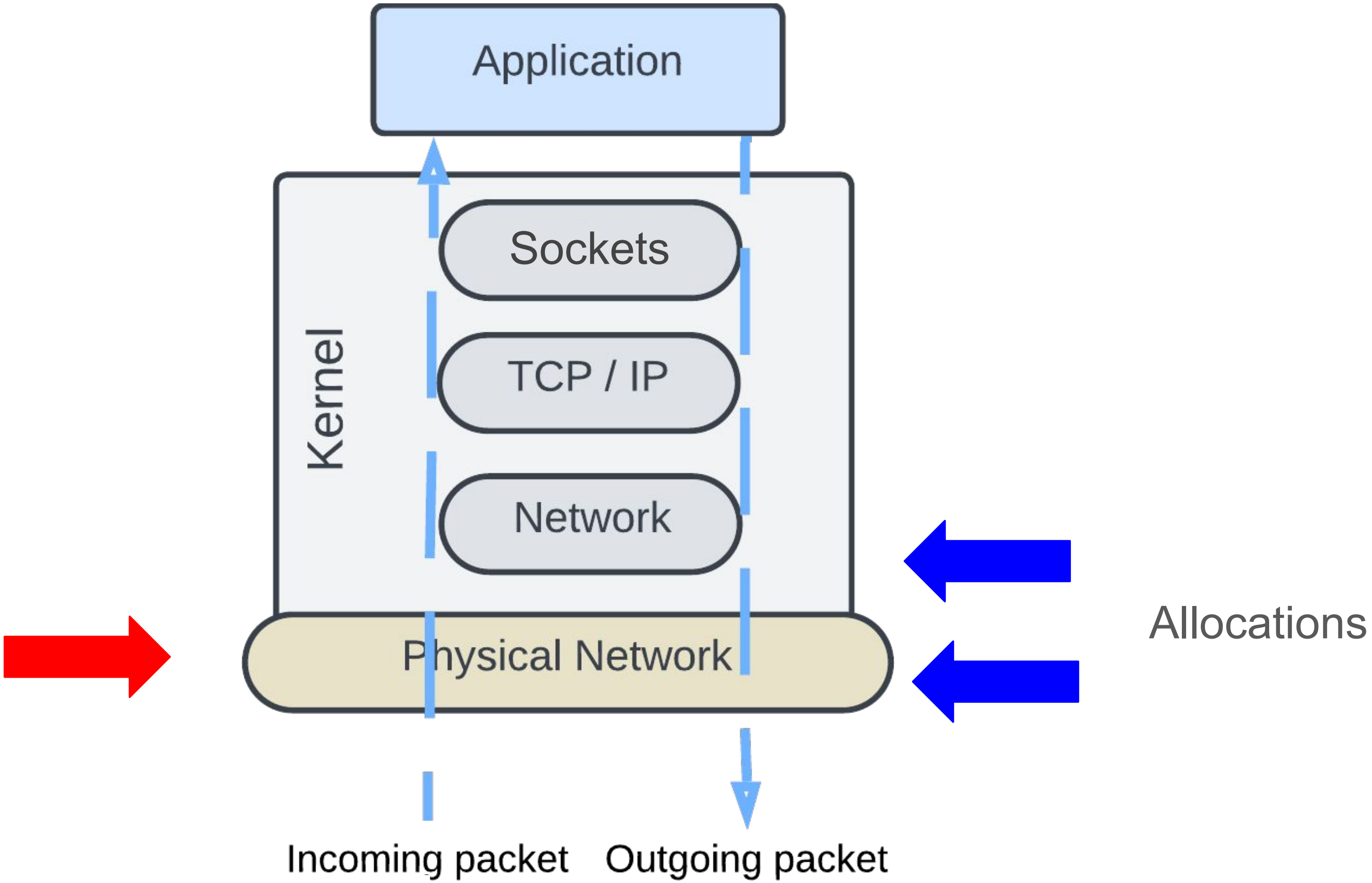
TCP / IP

Network

Physical Network

Incoming packet

Outgoing packet



Application

Kernel

Sockets

TCP / IP

Network

Physical Network

Allocations

Incoming packet

Outgoing packet

Any Ideas?

Traditional ways

Option 1: Change Kernel

Option 2: Kernel Module

“ You think you want a stable kernel interface, but you really do not, and you don’t even know it.

Greg Kroah-Hartman

https://www.kernel.org/doc/html/latest/process/stable-api-nonsense.html
https://www.youtube.com/watch?v=agC5N9I6jRE

Traditional ways

Option 1: Change Kernel

Option 2: Kernel Module

What about a third option?

eBPF

eBPF is making the Linux Kernel programmable at native execution speed!

> “eBPF is a crazy technology, it’s like putting JavaScript into the Linux kernel

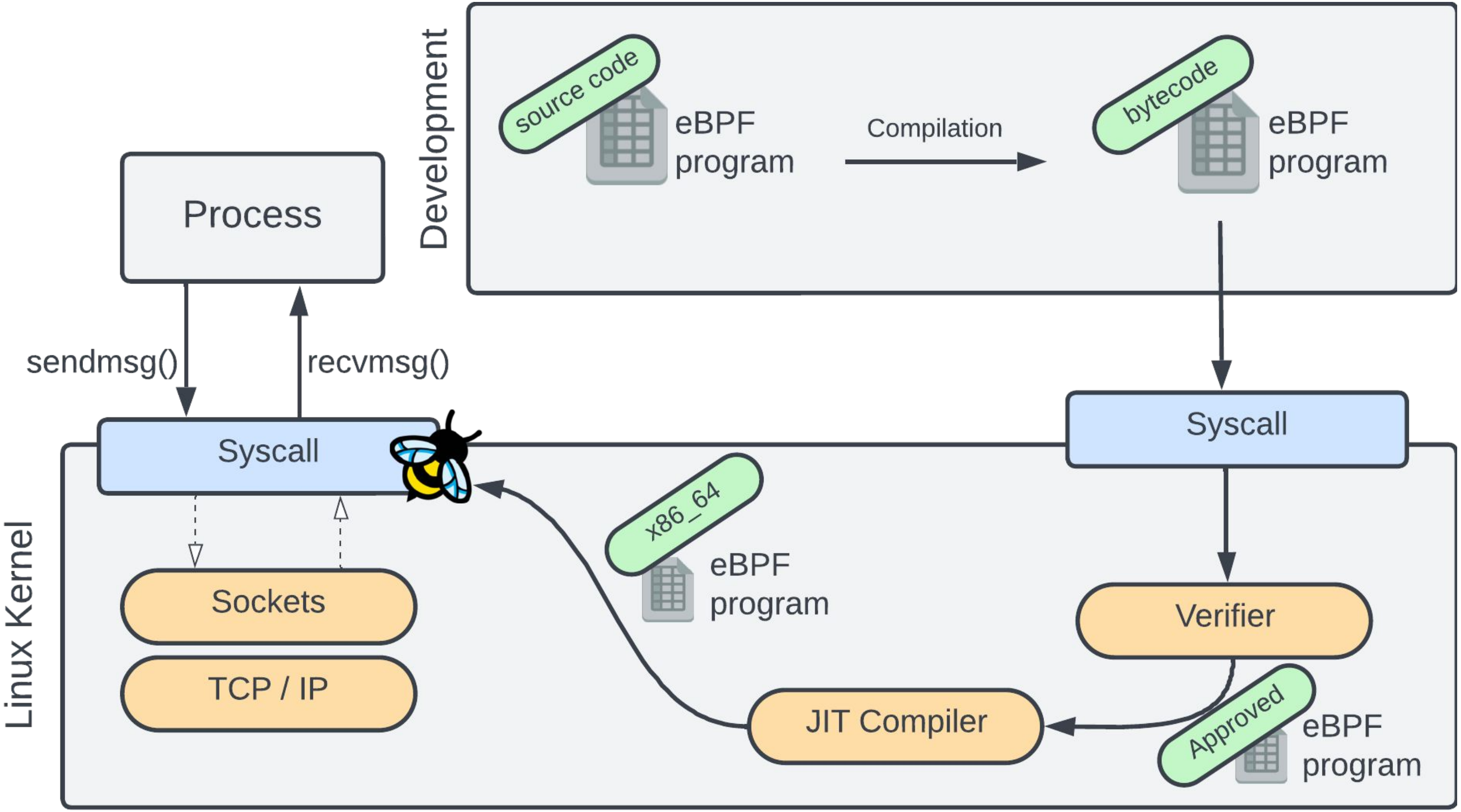
Brendan Gregg

https://www.youtube.com/watch?v=tDacjrSCeq4

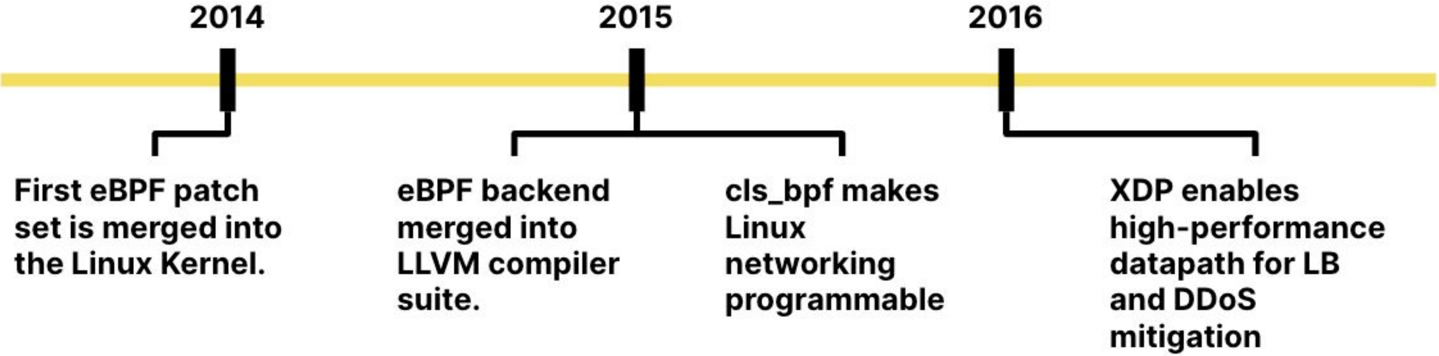
> “eBPF is a crazy technology, it’s like putting JavaScript into the Linux kernel
>
> Brendan Gregg

https://www.facesofopensource.com/brendan-gregg/

eBPF runtime



eBPF



2014

First eBPF patch set is merged into the Linux Kernel.

2015

eBPF backend merged into LLVM compiler suite.

cls_bpf makes Linux networking programmable

2016

XDP enables high-performance datapath for LB and DDoS mitigation

https://ebpf.io/blog/2023/11/03/the-ebpf-documentary/

https://cilium.io/blog/2020/11/10/ebpf-future-of-networking/

eBPF runtime

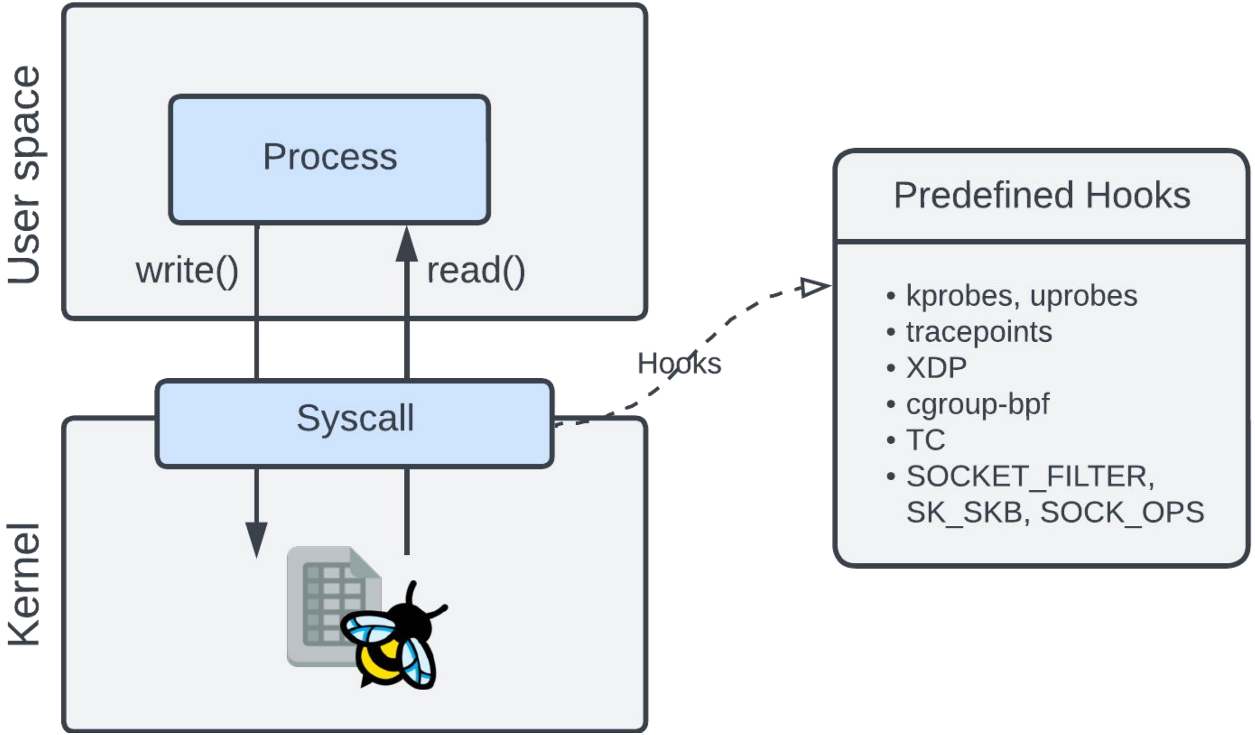
Safety and Security

Continuous delivery

Efficiency

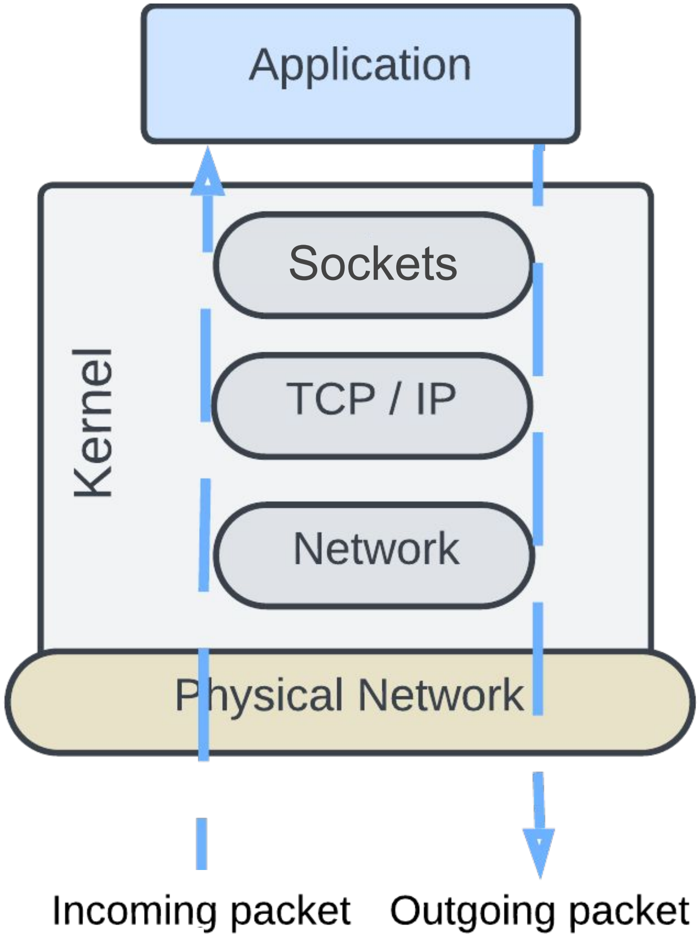
Standard

eBPF hooks



Predefined Hooks

- kprobes, uprobes
- tracepoints
- XDP
- cgroup-bpf
- TC
- SOCKET_FILTER, SK_SKB, SOCK_OPS



Application

Kernel

Sockets

TCP / IP

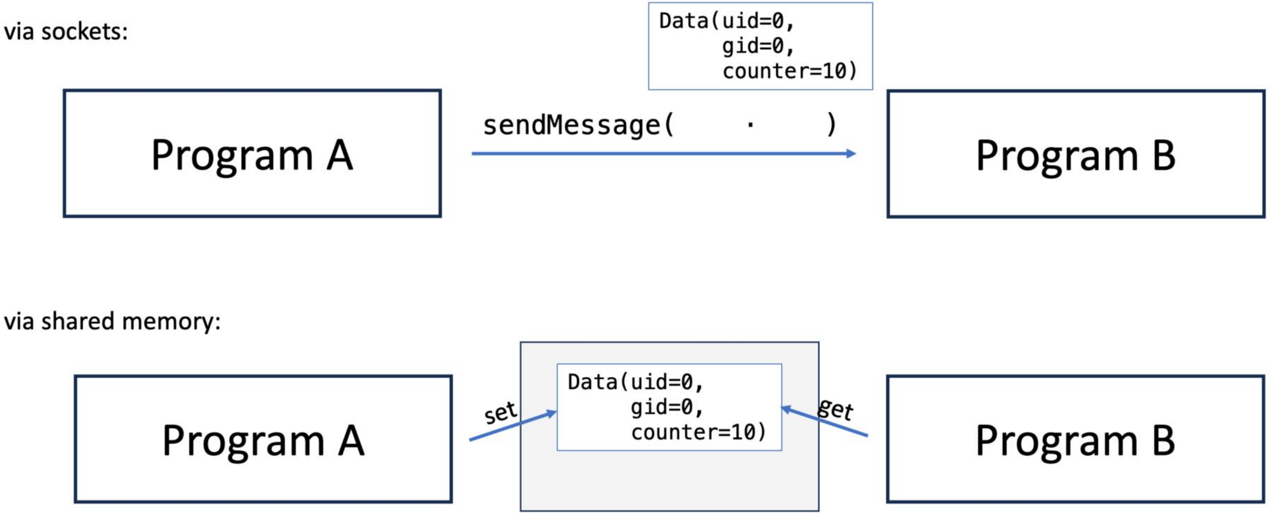
Network

Physical Network

Incoming packet

Outgoing packet

How to share data?



via sockets:

Program A

sendMessage(·)

```
Data(uid=0,
     gid=0,
     counter=10)
```

Program B

via shared memory:

Program A

set

```
Data(uid=0,
     gid=0,
     counter=10)
```

get

Program B

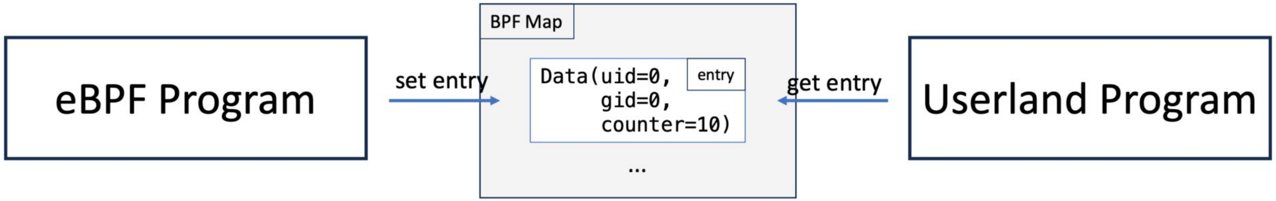
https://mostlynerdless.de/blog/2024/01/12/hello-ebpf-recording-data-in-basic-ebpf-maps-2/

How to share data?

Any Problems?

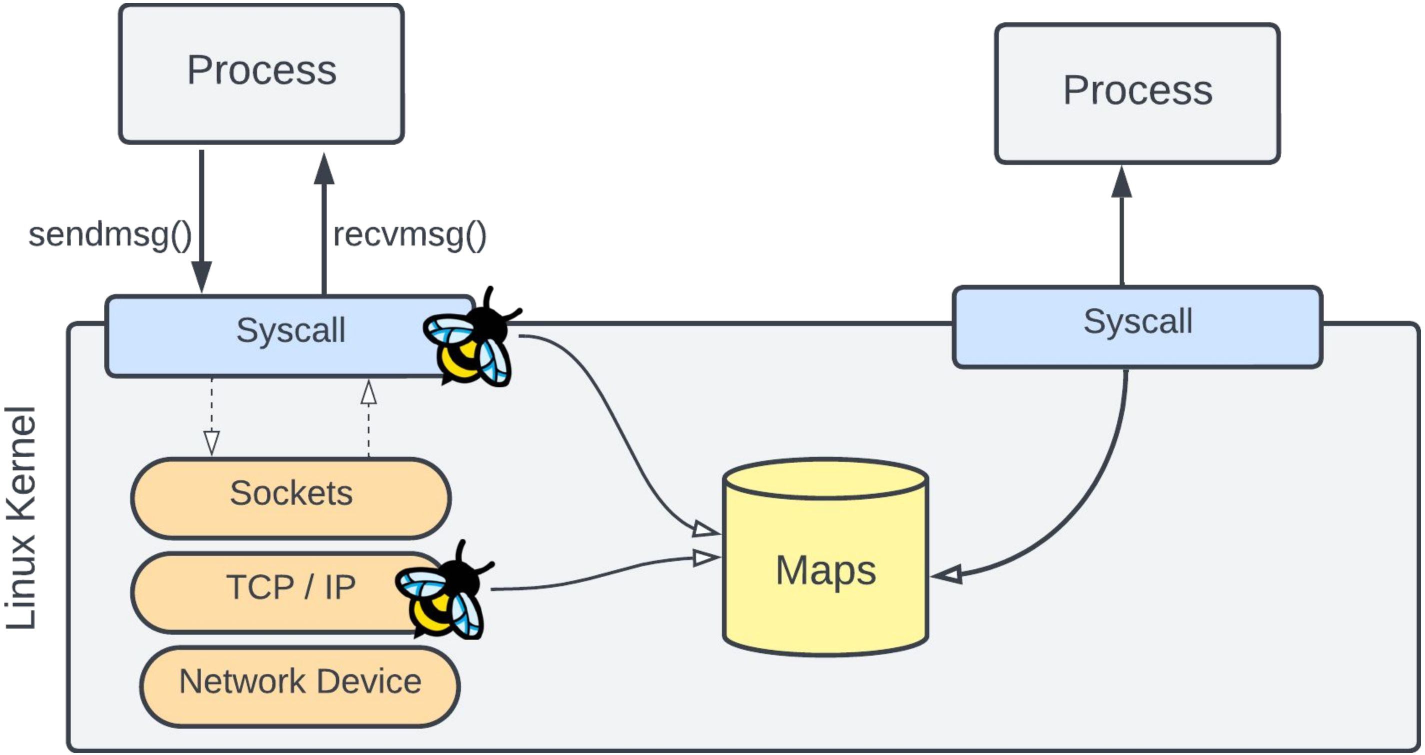
How to share data?

via eBPF maps:



https://mostlynerdless.de/blog/2024/01/12/hello-ebpf-recording-data-in-basic-ebpf-maps-2/

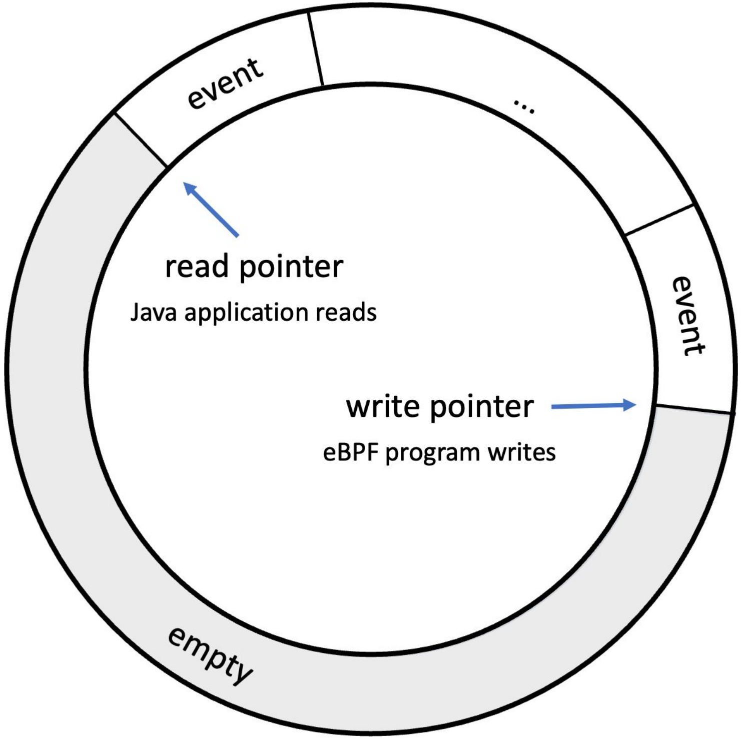
eBPF Maps


Process
sendmsg()
recvmsg()
Syscall
Linux Kernel
Sockets
TCP / IP
Network Device
Maps
Process
Syscall

eBPF Maps

Map Types

- Hash tables, Arrays
- Perf and Ring Buffers
- LPM trie Maps
- LRU Maps
- Queue & Stack Maps
- Stack Trace Maps
- …

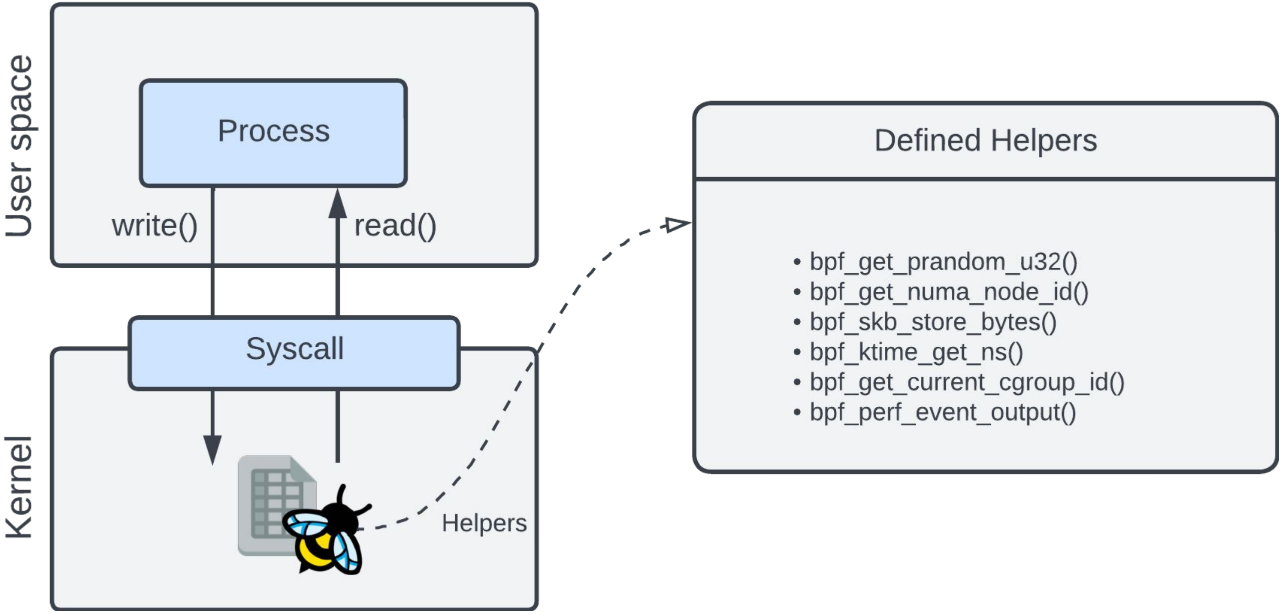


Ring Buffer for event

One for all CPUs

https://mostlynerdless.de/blog/2024/03/12/hello-ebpf-ring-buffers-in-libbpf-6/

eBPF Helpers

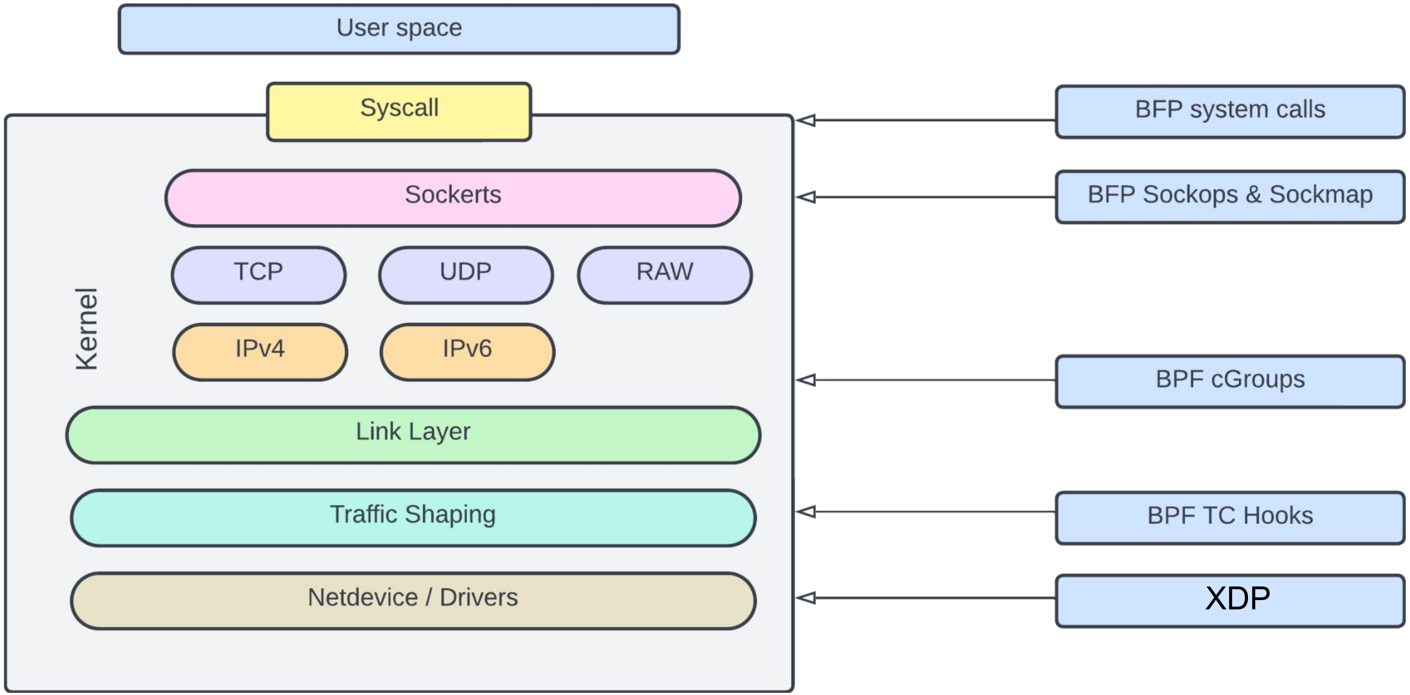


Defined Helpers

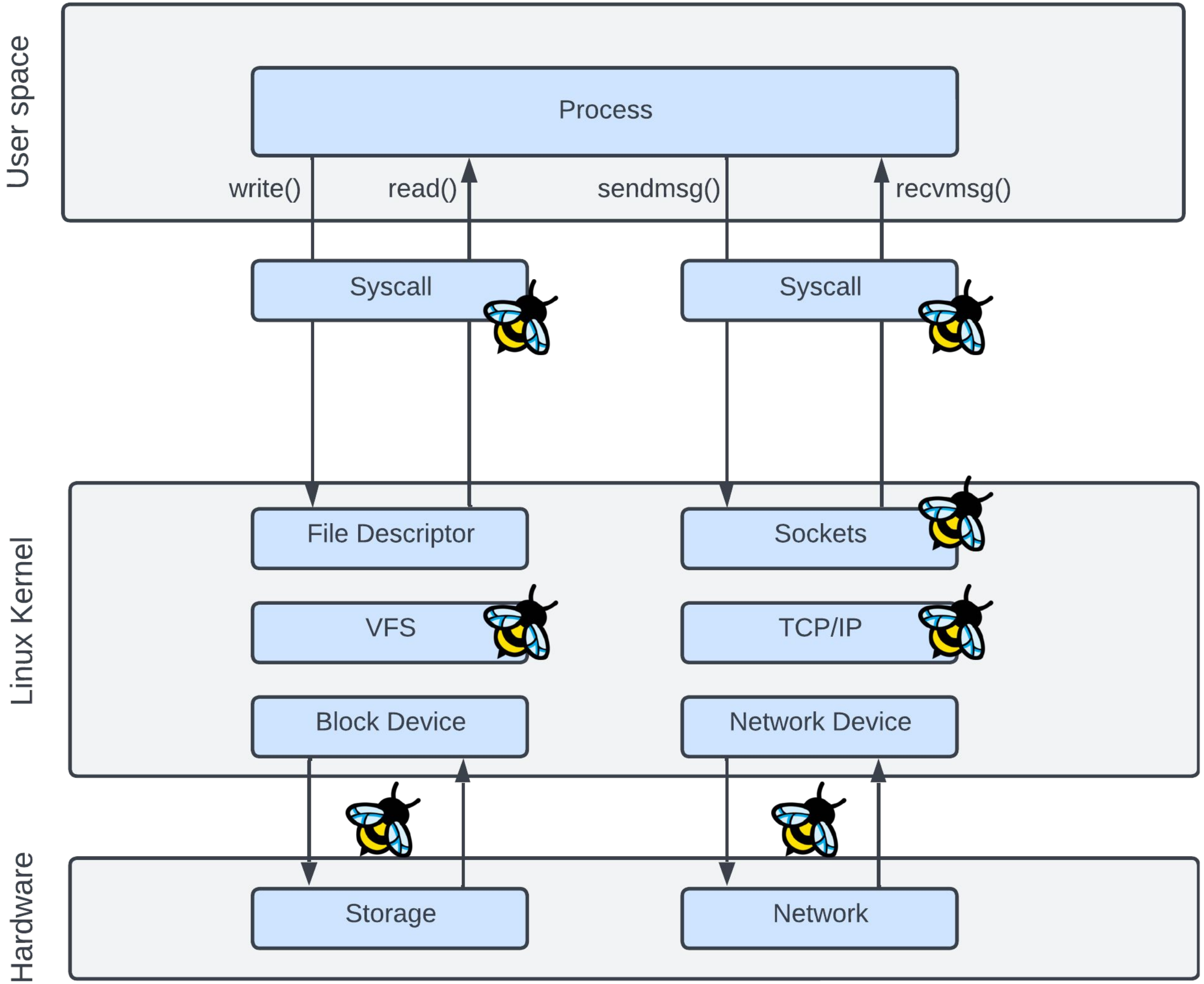
- bpf_get_prandom_u32()
- bpf_get_numa_node_id()
- bpf_skb_store_bytes()
- bpf_ktime_get_ns()
- bpf_get_current_cgroup_id()
- bpf_perf_event_output()

https://github.com/torvalds/linux/blob/master/kernel/bpf/helpers.c

eBPF use-cases



eBPF and networking

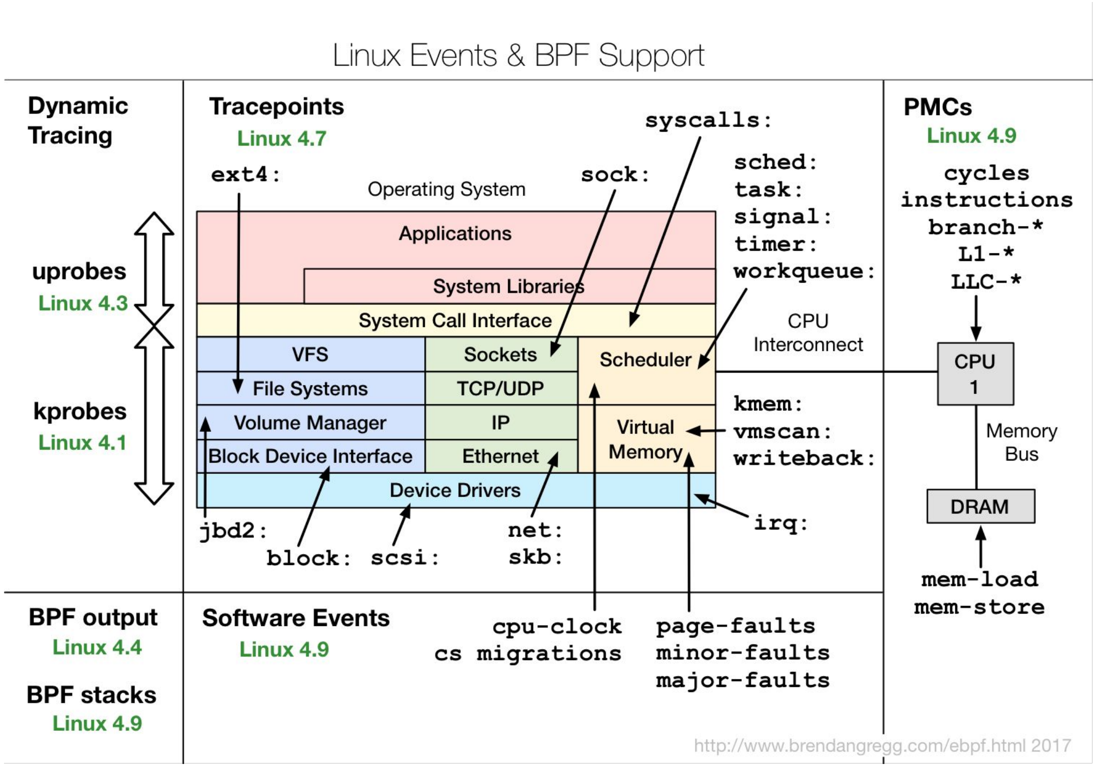


https://ebpf.io/what-is-ebpf/

eBPF use-cases

Tracing and Profiling

Everything is visible

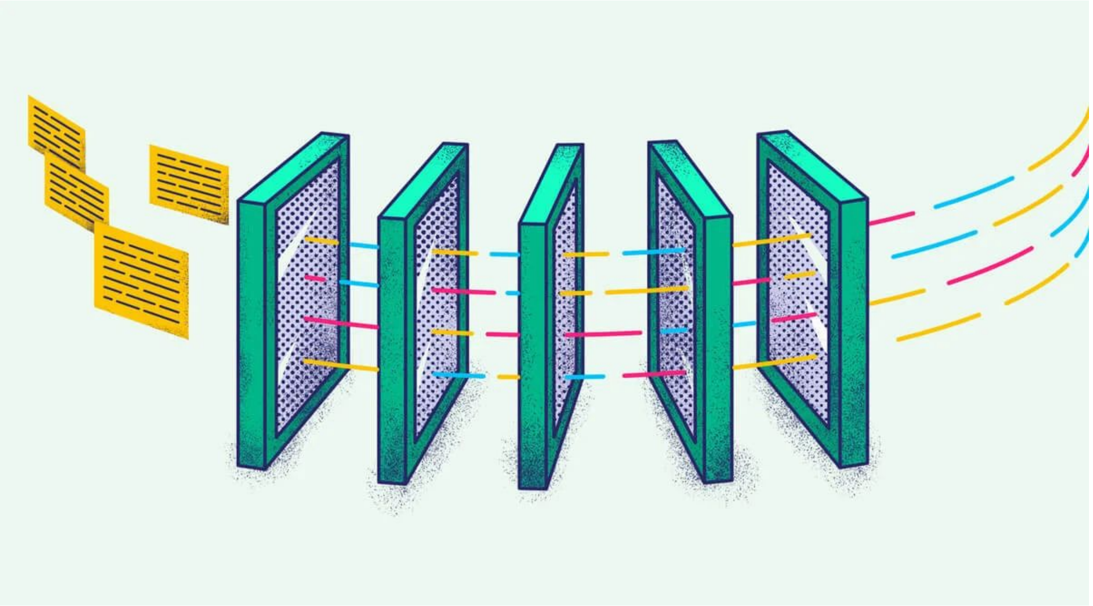


https://www.brendangregg.com/ebpf.html

eBPF use-cases

Observability and Monitoring

Observability with eBPF is secure, isolated, and non-obtrusive and can be exported to centralized platforms.



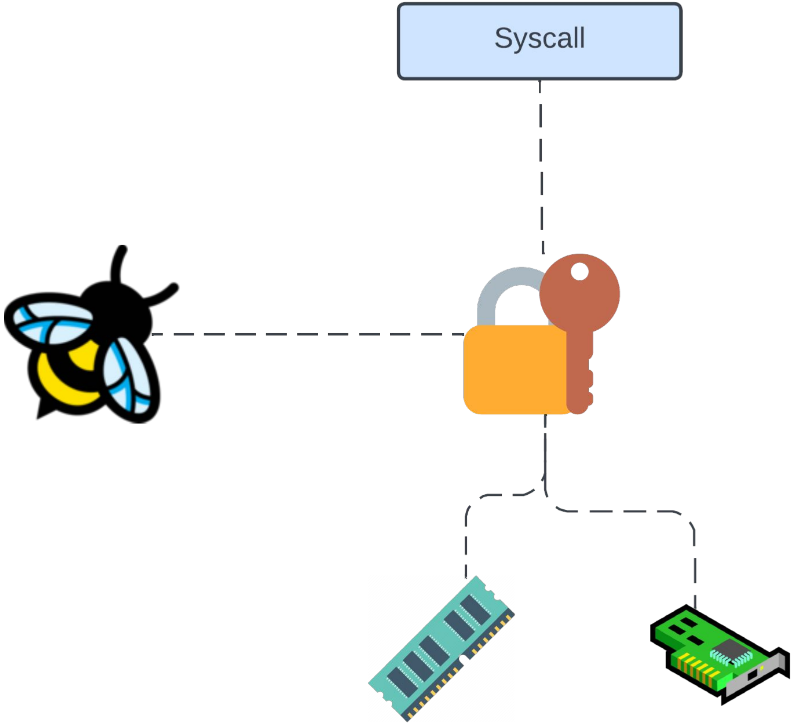
https://coralogix.com/blog/what-ebpf-why-important-for-observability/

eBPF use-cases

Security Control

- eBPF facilitates the combination of control and visibility over all aspects

- Possibility to build security systems that operate with more context and an improved level of control.



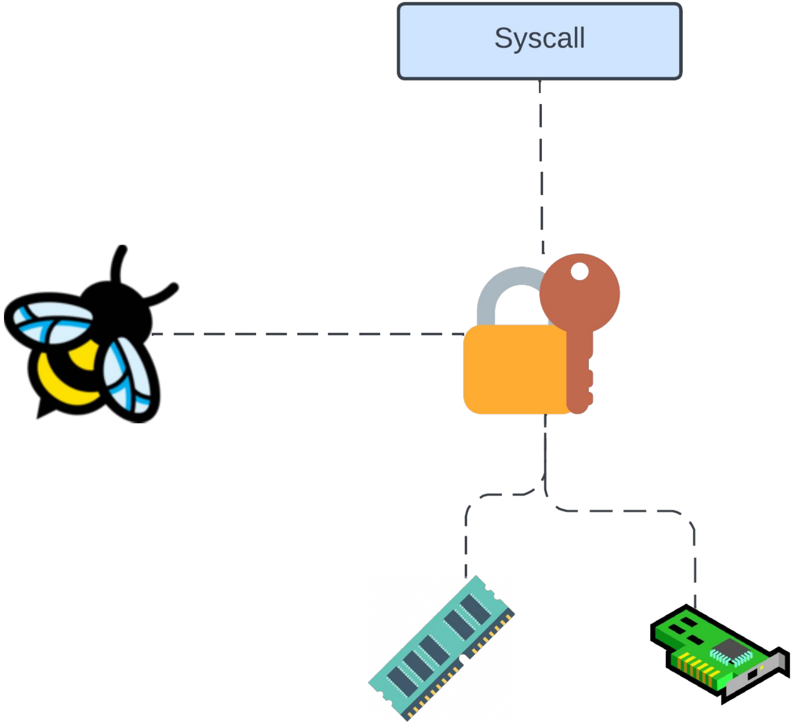
https://coralogix.com/blog/what-ebpf-why-important-for-observability/

eBPF use-cases

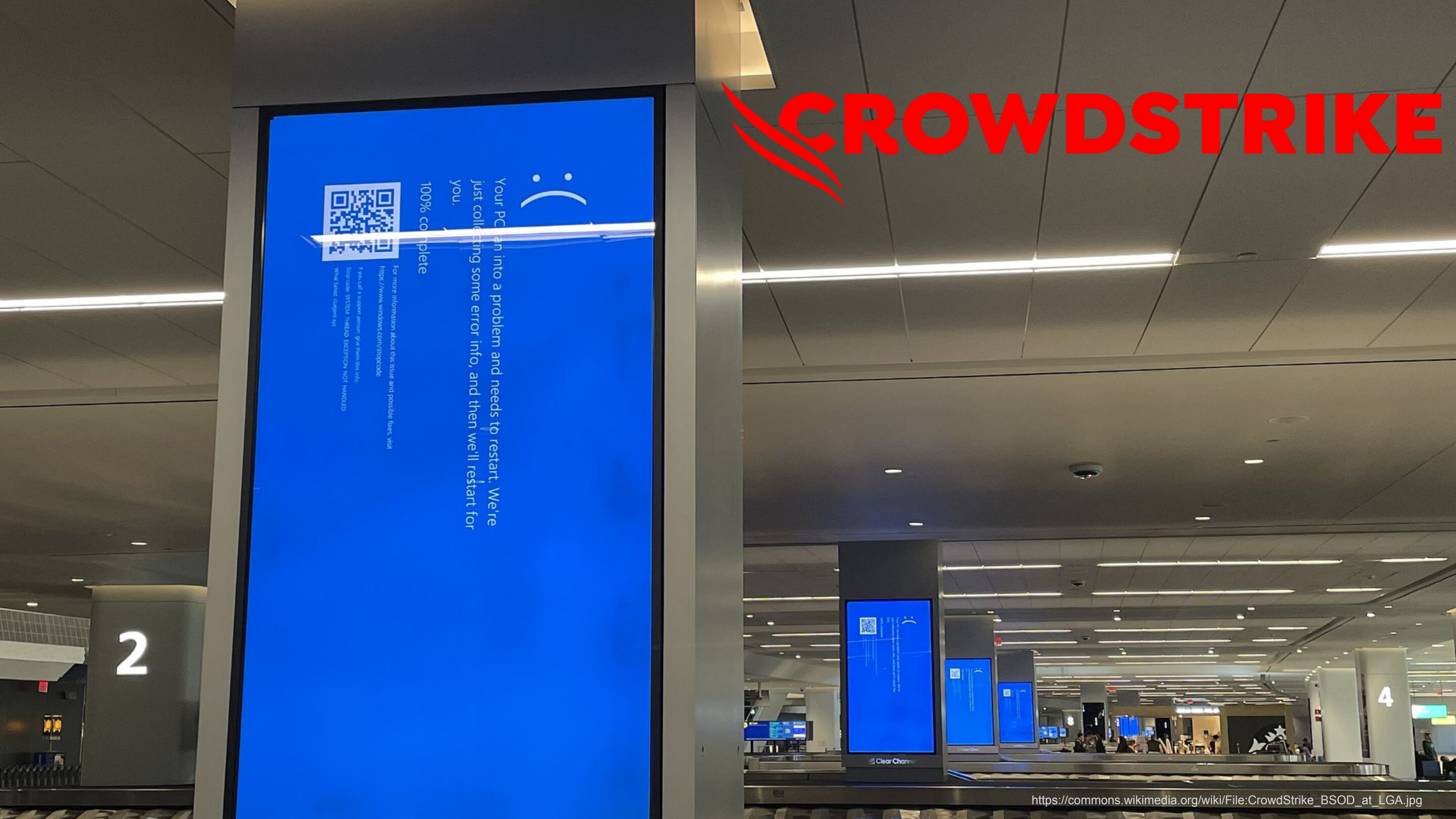
Security Control

- eBPF facilitates the combination of control and visibility over all aspects

- Possibility to build security systems that operate with more context and an improved level of control.

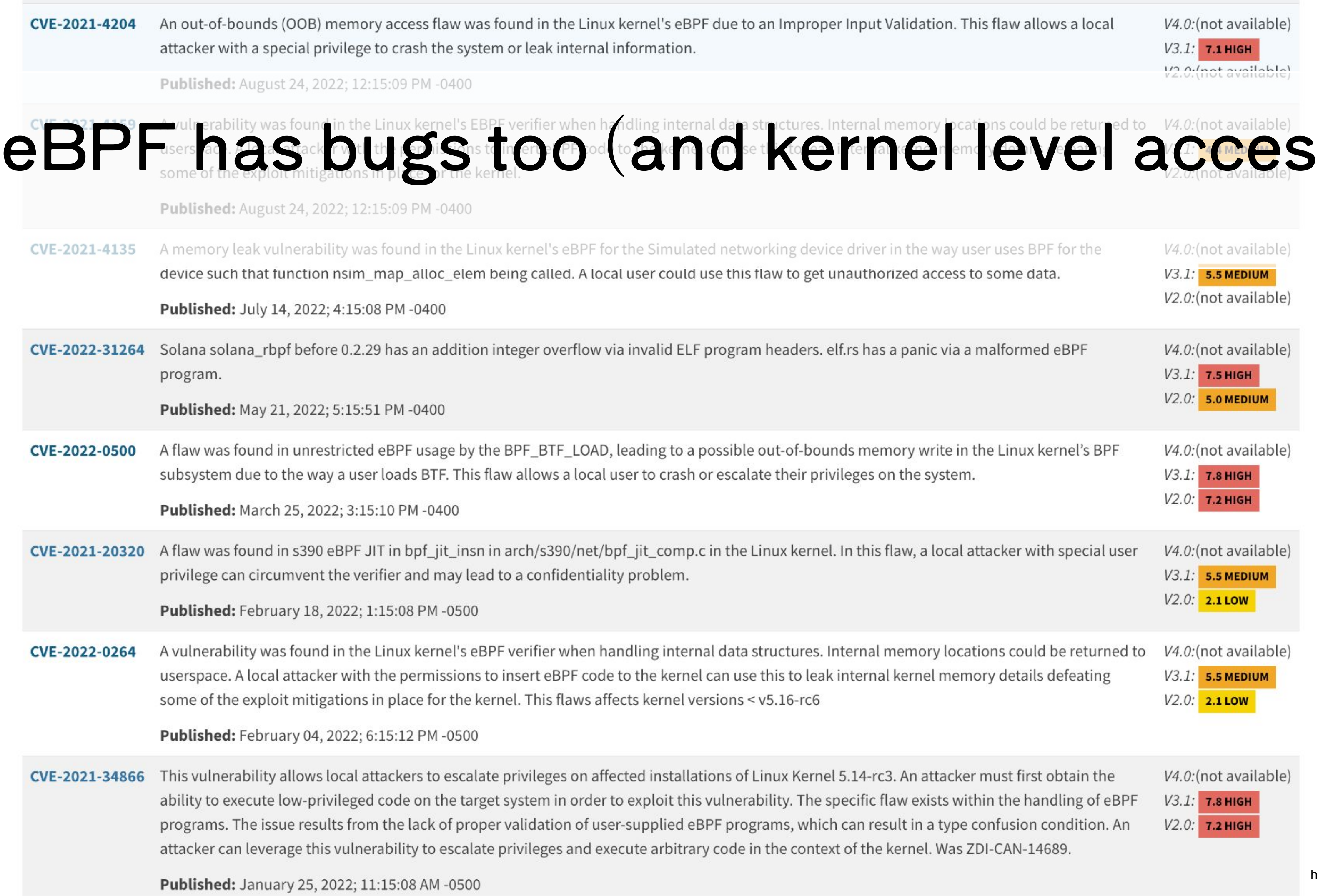


https://coralogix.com/blog/what-ebpf-why-important-for-observability/


CROWDSTRIKE
:(
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.
100% complete
For more information about this issue and possible fixes, visit https://www.windows.com/stopcode
If you call a support person, give them this info:
Stop code: SYSTEM THREAD EXCEPTION NOT HANDLED
What failed: csagent.sys
2
4
Clear Channel

https://commons.wikimedia.org/wiki/File:CrowdStrike_BSOD_at_LGA.jpg

eBPF has bugs too (and kernel level access)



CVE-2021-4204 An out-of-bounds (OOB) memory access flaw was found in the Linux kernel's eBPF due to an Improper Input Validation. This flaw allows a local attacker with a special privilege to crash the system or leak internal information.

Published: August 24, 2022; 12:15:09 PM -0400

V4.0:(not available)
V3.1: **7.1 HIGH**
V2.0:(not available)

CVE-2021-4159 A vulnerability was found in the Linux kernel's EBPF verifier when handling internal data structures. Internal memory locations could be returned to userspace. A local attacker with the permissions to insert eBPF code to the kernel can use this to leak internal kernel memory details defeating some of the exploit mitigations in place for the kernel.

Published: August 24, 2022; 12:15:09 PM -0400

V4.0:(not available)
V3.1: **4.4 MEDIUM**
V2.0:(not available)

CVE-2021-4135 A memory leak vulnerability was found in the Linux kernel's eBPF for the Simulated networking device driver in the way user uses BPF for the device such that function nsim_map_alloc_elem being called. A local user could use this flaw to get unauthorized access to some data.

Published: July 14, 2022; 4:15:08 PM -0400

V4.0:(not available)
V3.1: **5.5 MEDIUM**
V2.0:(not available)

CVE-2022-31264 Solana solana_rbpf before 0.2.29 has an addition integer overflow via invalid ELF program headers. elf.rs has a panic via a malformed eBPF program.

Published: May 21, 2022; 5:15:51 PM -0400

V4.0:(not available)
V3.1: **7.5 HIGH**
V2.0: **5.0 MEDIUM**

CVE-2022-0500 A flaw was found in unrestricted eBPF usage by the BPF_BTF_LOAD, leading to a possible out-of-bounds memory write in the Linux kernel's BPF subsystem due to the way a user loads BTF. This flaw allows a local user to crash or escalate their privileges on the system.

Published: March 25, 2022; 3:15:10 PM -0400

V4.0:(not available)
V3.1: **7.8 HIGH**
V2.0: **7.2 HIGH**

CVE-2021-20320 A flaw was found in s390 eBPF JIT in bpf_jit_insn in arch/s390/net/bpf_jit_comp.c in the Linux kernel. In this flaw, a local attacker with special user privilege can circumvent the verifier and may lead to a confidentiality problem.

Published: February 18, 2022; 1:15:08 PM -0500

V4.0:(not available)
V3.1: **5.5 MEDIUM**
V2.0: **2.1 LOW**

CVE-2022-0264 A vulnerability was found in the Linux kernel's eBPF verifier when handling internal data structures. Internal memory locations could be returned to userspace. A local attacker with the permissions to insert eBPF code to the kernel can use this to leak internal kernel memory details defeating some of the exploit mitigations in place for the kernel. This flaws affects kernel versions < v5.16-rc6

Published: February 04, 2022; 6:15:12 PM -0500

V4.0:(not available)
V3.1: **5.5 MEDIUM**
V2.0: **2.1 LOW**

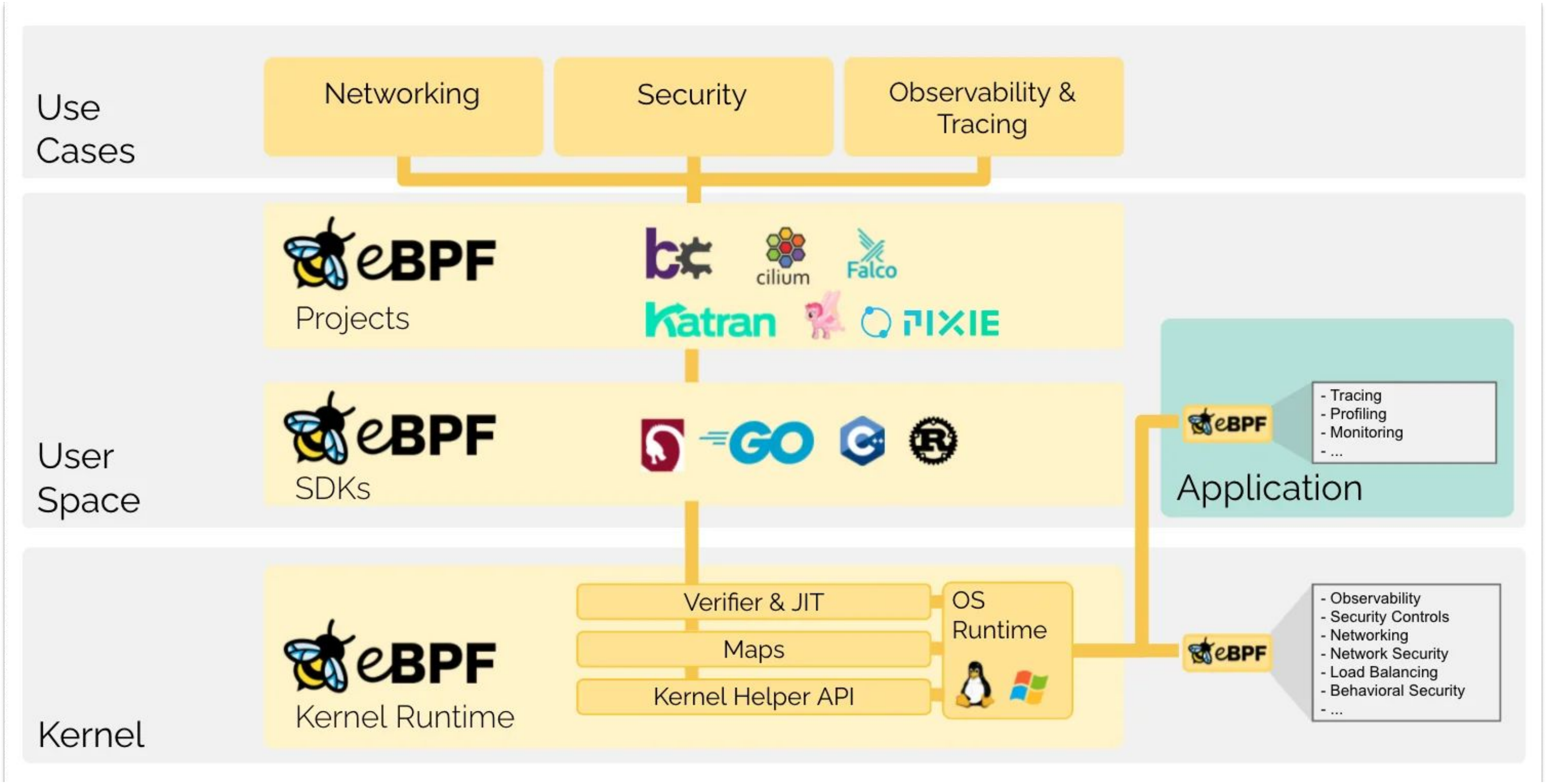
CVE-2021-34866 This vulnerability allows local attackers to escalate privileges on affected installations of Linux Kernel 5.14-rc3. An attacker must first obtain the ability to execute low-privileged code on the target system in order to exploit this vulnerability. The specific flaw exists within the handling of eBPF programs. The issue results from the lack of proper validation of user-supplied eBPF programs, which can result in a type confusion condition. An attacker can leverage this vulnerability to escalate privileges and execute arbitrary code in the context of the kernel. Was ZDI-CAN-14689.

Published: January 25, 2022; 11:15:08 AM -0500

V4.0:(not available)
V3.1: **7.8 HIGH**
V2.0: **7.2 HIGH**

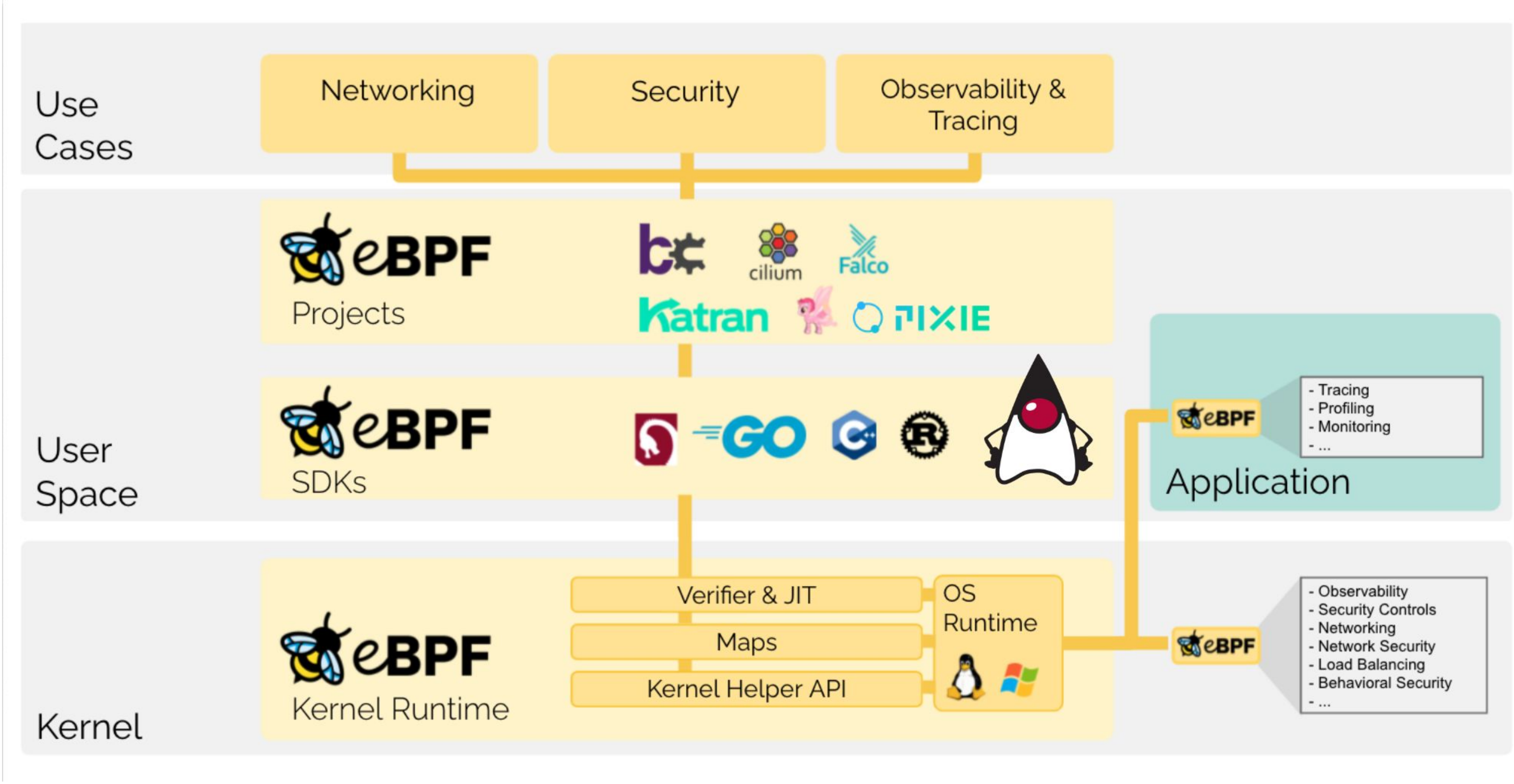
https://nvd.nist.gov/vuln/

eBPF Ecosystem



https://ebpf.io

eBPF Ecosystem



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

“

eBPF is a crazy technology, it’s like putting JavaScript into the Linux kernel

Brendan Gregg

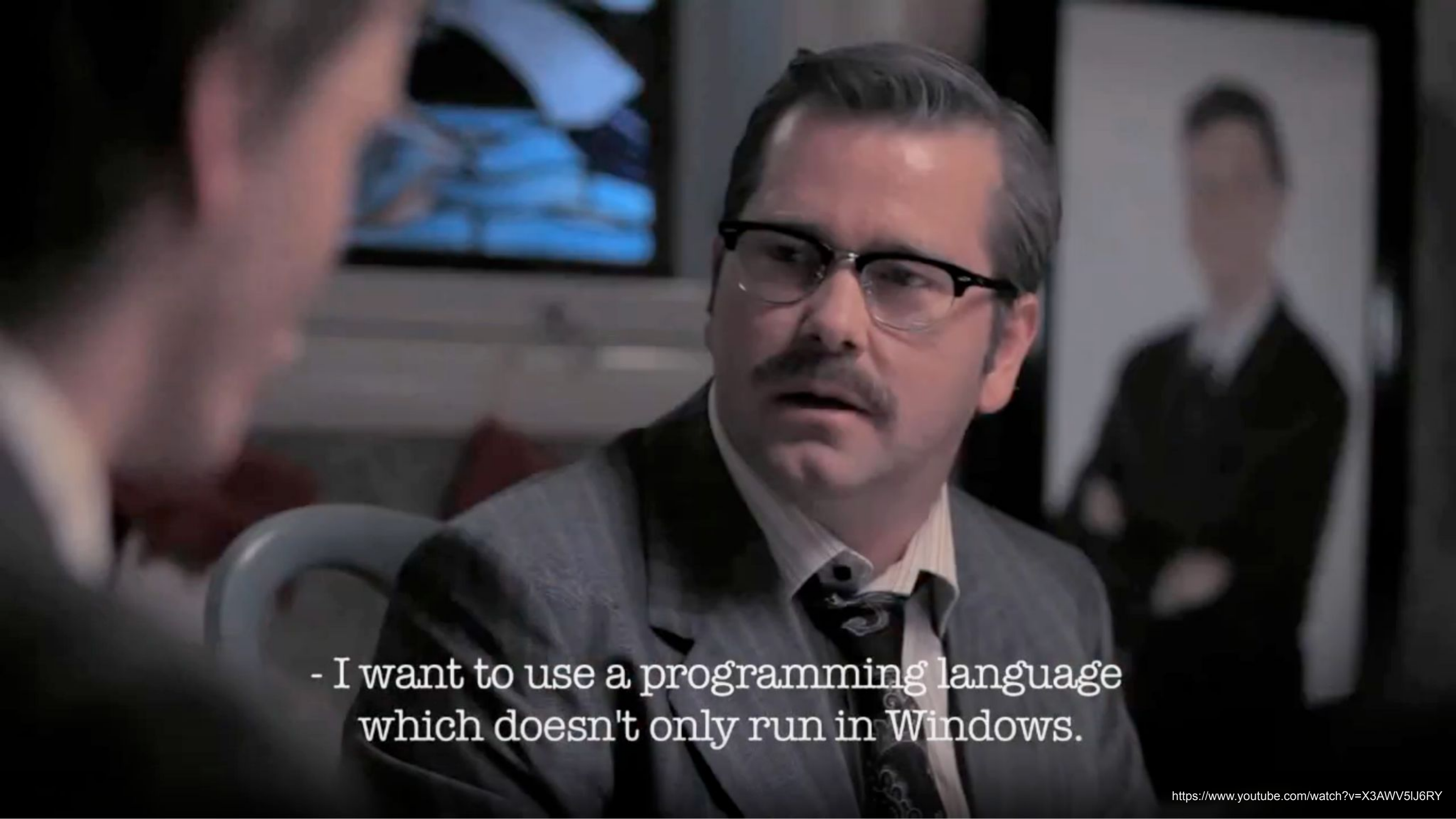
https://www.youtube.com/watch?v=tDacjrSCeq4

“

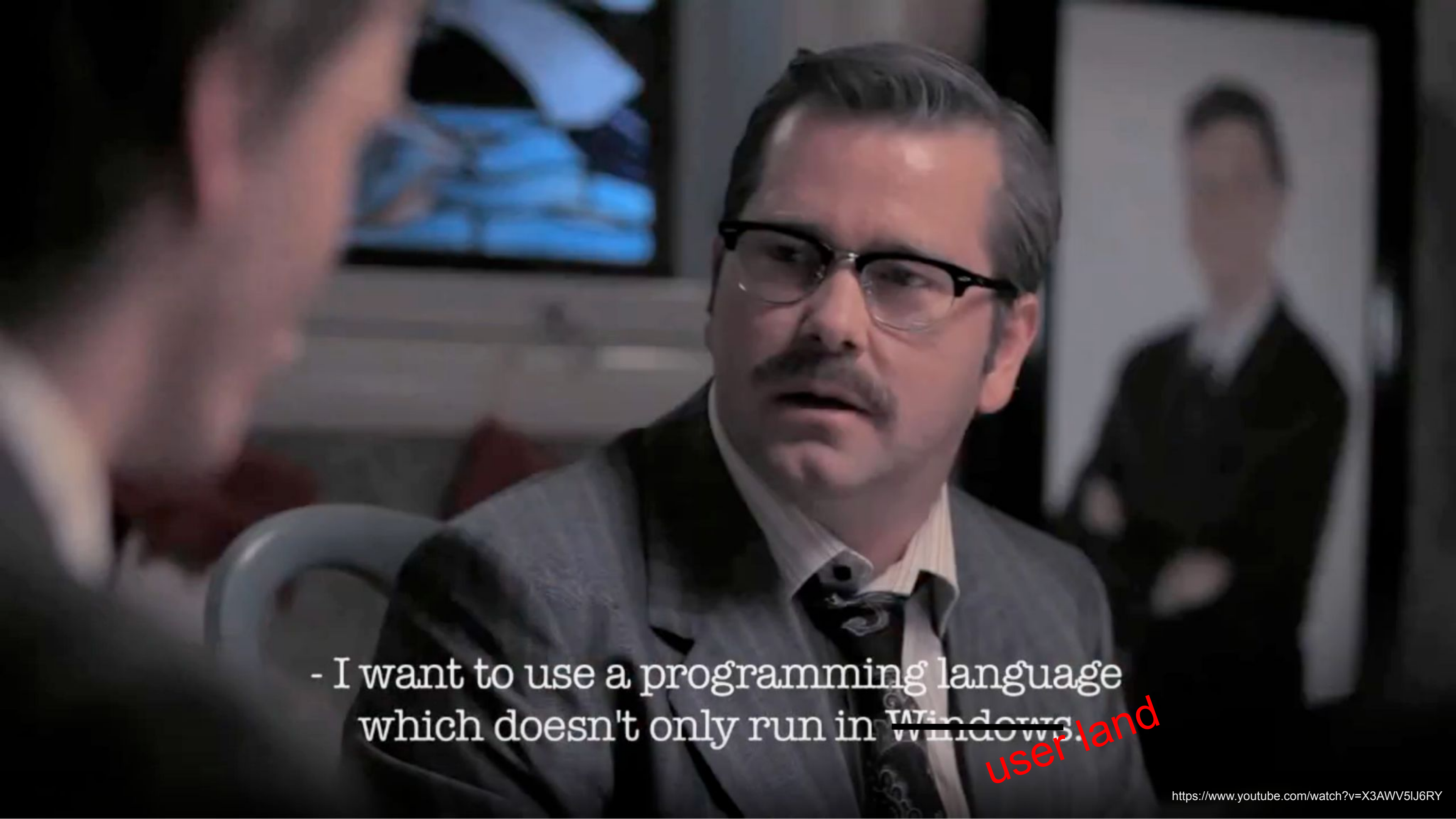
eBPF is a crazy technology, it’s like putting Java~~Script~~ into the Linux kernel

Brendan Gregg

https://www.youtube.com/watch?v=tDacjrSCeq4


- I want to use a programming language
which doesn't only run in Windows.

https://www.youtube.com/watch?v=X3AWV5lJ6RY



- I want to use a programming language which doesn't only run in ~~Windows~~.

user land

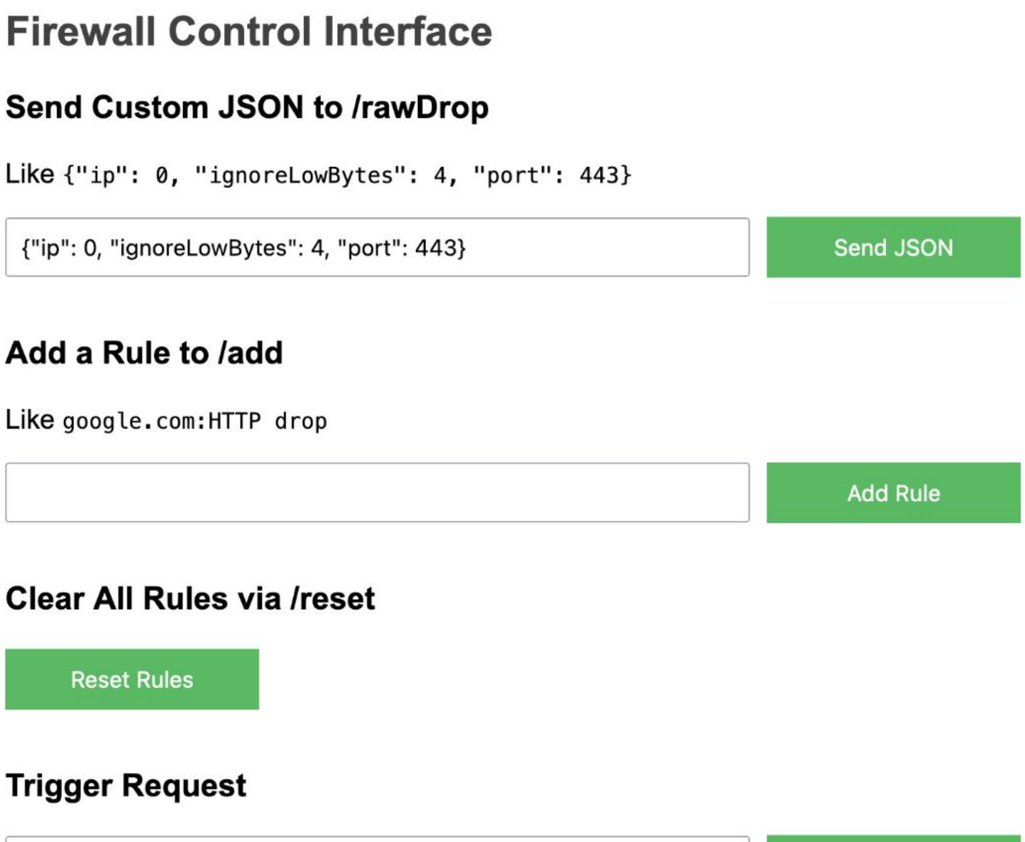
https://www.youtube.com/watch?v=X3AWV5lJ6RY

hello eBPF

helloeBPF

Work in Progress

Firewall Demo



Firewall Control Interface

Send Custom JSON to /rawDrop

Like `{"ip": 0, "ignoreLowBytes": 4, "port": 443}`

{"ip": 0, "ignoreLowBytes": 4, "port": 443}

Add a Rule to /add

Like `google.com:HTTP drop`

Clear All Rules via /reset

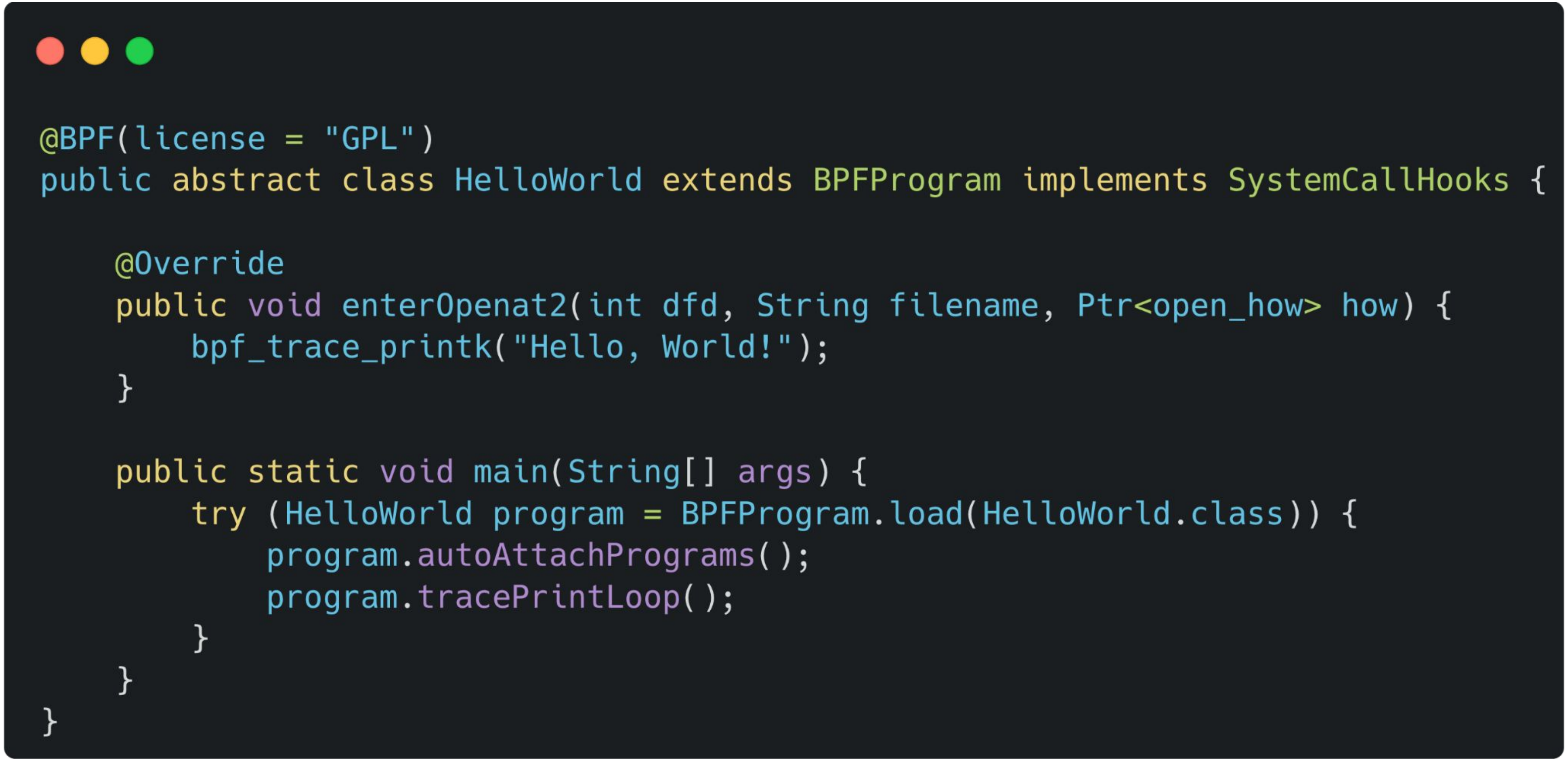
Trigger Request

Live Coding

Live Coding

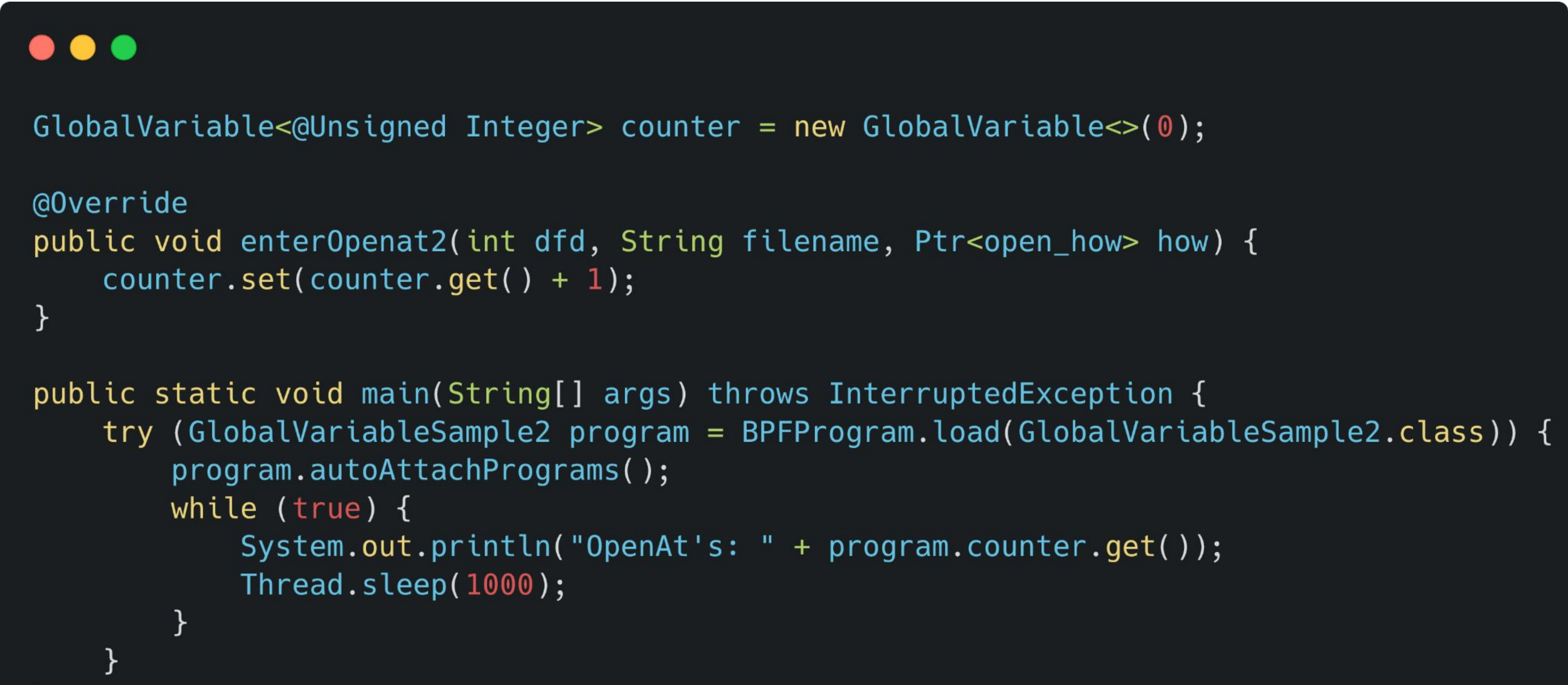
Having fun with eBPF

Hello World!



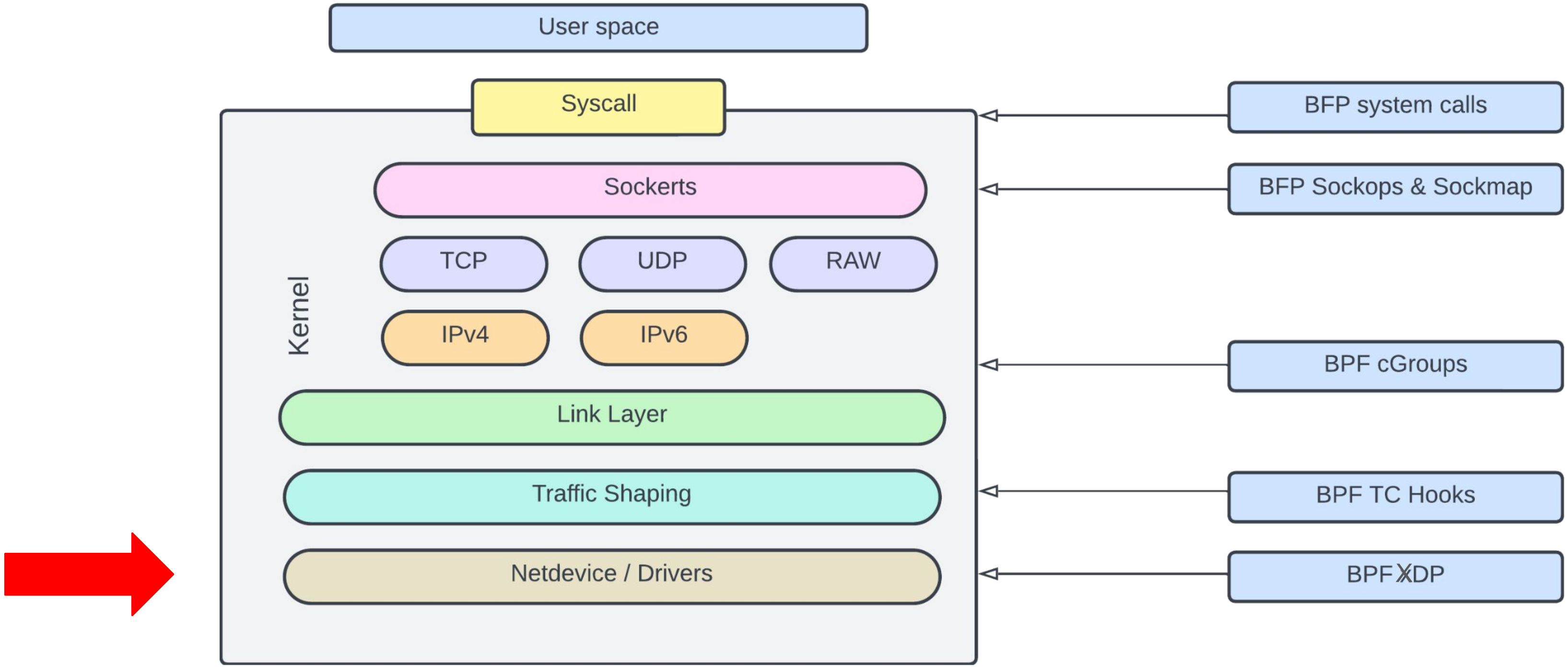
```java
@BPF(license = "GPL")
public abstract class HelloWorld extends BPFProgram implements SystemCallHooks {

    @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)) {
            program.autoAttachPrograms();
            program.tracePrintLoop();
        }
    }
}
```

Global Variables



```
GlobalVariable<@Unsigned Integer> counter = new GlobalVariable<>(0);

@Override
public void enterOpenat2(int dfd, String filename, Ptr<open_how> how) {
    counter.set(counter.get() + 1);
}

public static void main(String[] args) throws InterruptedException {
    try (GlobalVariableSample2 program = BPFProgram.load(GlobalVariableSample2.class)) {
        program.autoAttachPrograms();
        while (true) {
            System.out.println("OpenAt's: " + program.counter.get());
            Thread.sleep(1000);
        }
    }
```

XDP



User space

Syscall

Kernel

Sockerts

TCP

UDP

RAW

IPv4

IPv6

Link Layer

Traffic Shaping

Netdevice / Drivers

BFP system calls

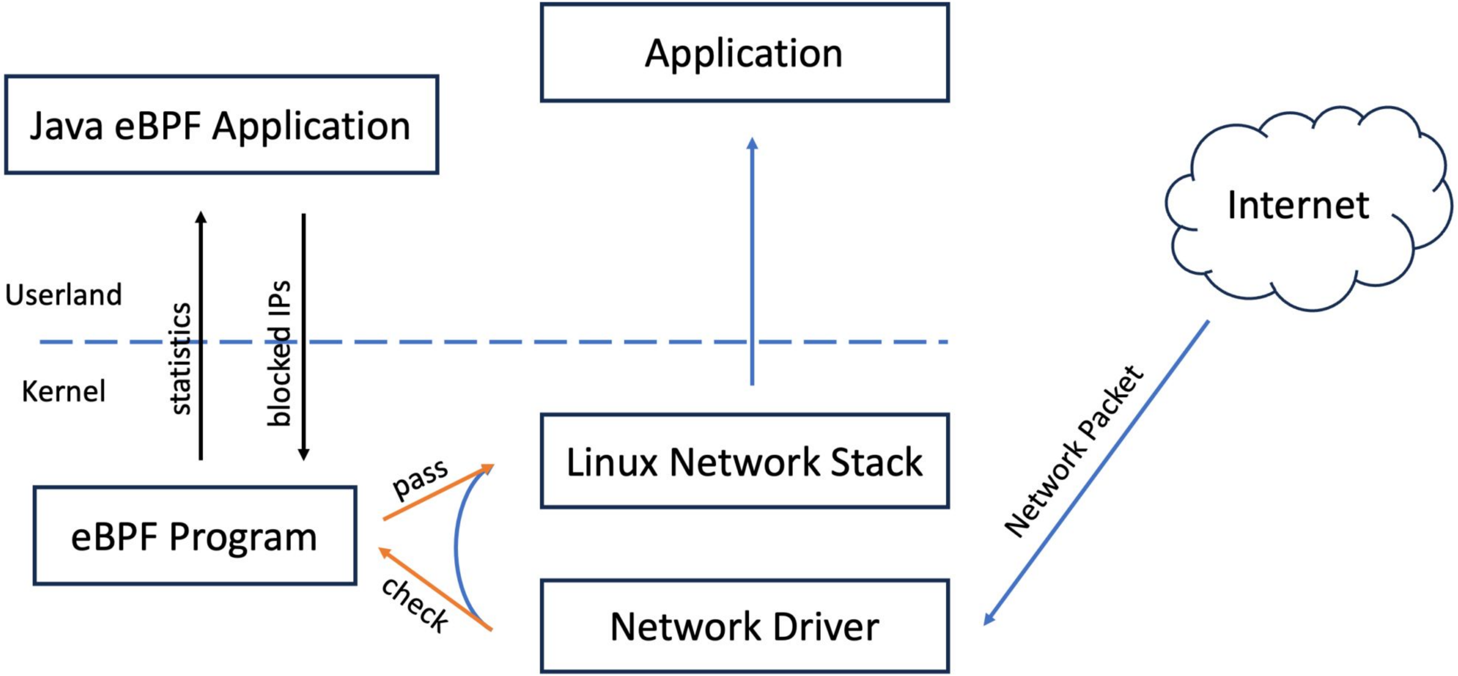
BFP Sockops & Sockmap

BPF cGroups

BPF TC Hooks

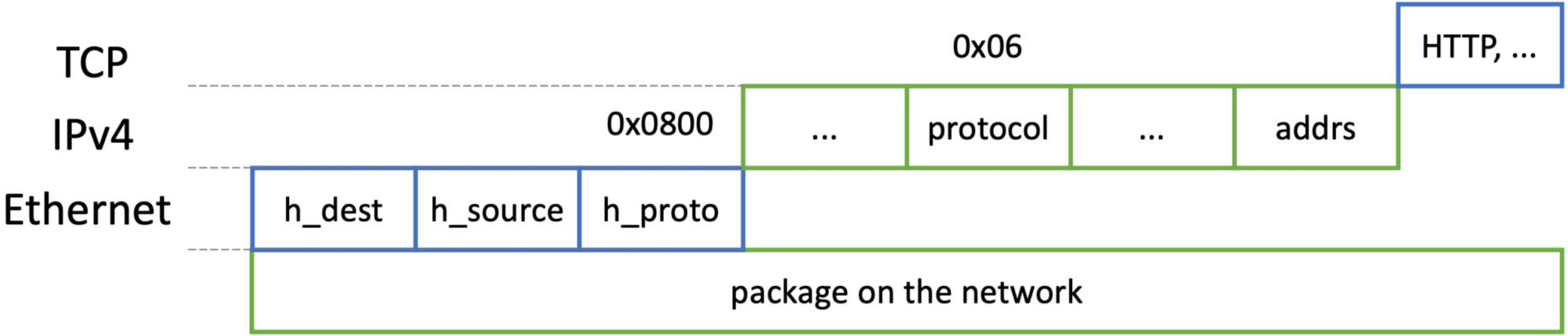
BPF XDP

XDP



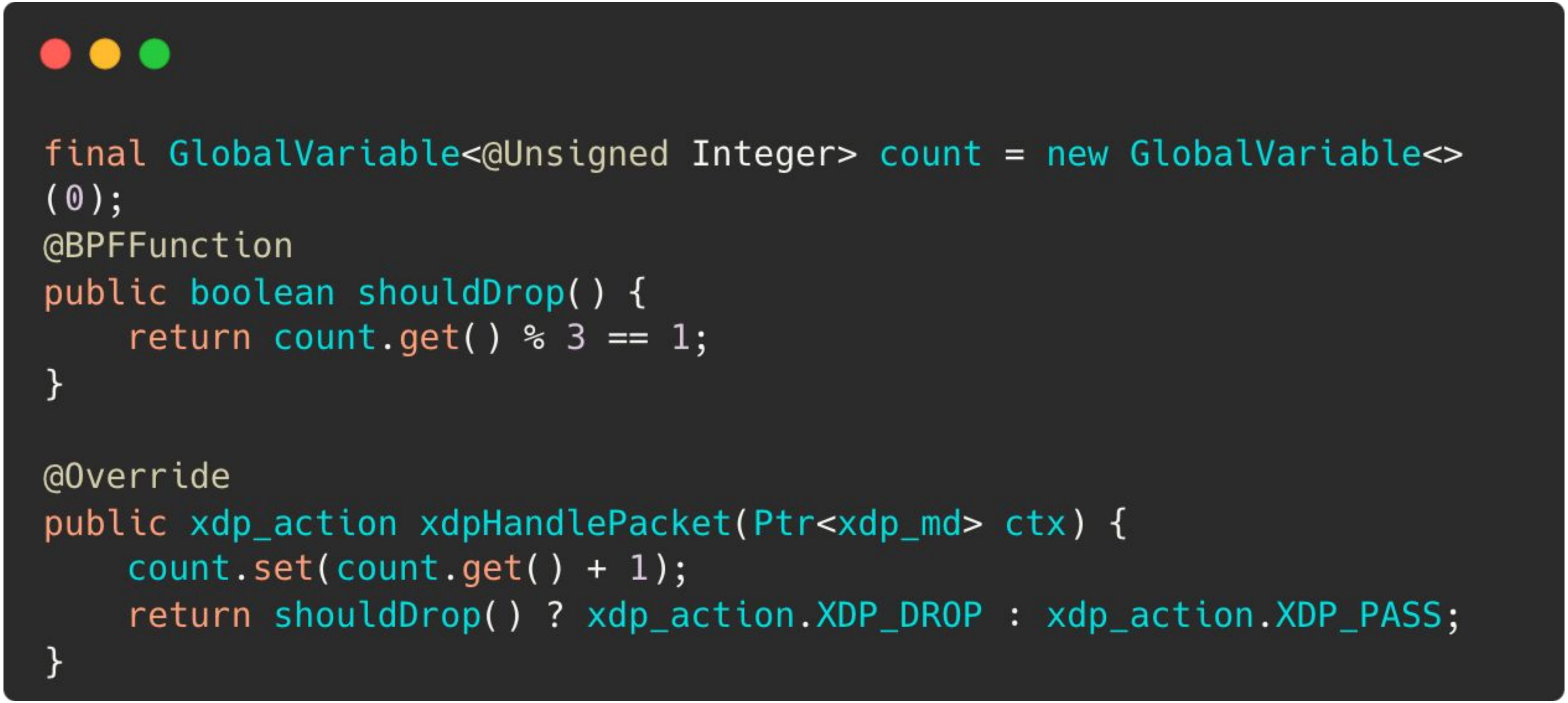
https://mostlynerdless.de/blog/2024/04/22/hello-ebpf-xdp-based-packet-filter-9

XDP



https://mostlynerdless.de/blog/2024/04/22/hello-ebpf-xdp-based-packet-filter-9

XDP



```
final GlobalVariable<@Unsigned Integer> count = new GlobalVariable<>
(0);
@BPFFunction
public boolean shouldDrop() {
    return count.get() % 3 == 1;
}

@Override
public xdp_action xdpHandlePacket(Ptr<xdp_md> ctx) {
    count.set(count.get() + 1);
    return shouldDrop() ? xdp_action.XDP_DROP : xdp_action.XDP_PASS;
}
```

“

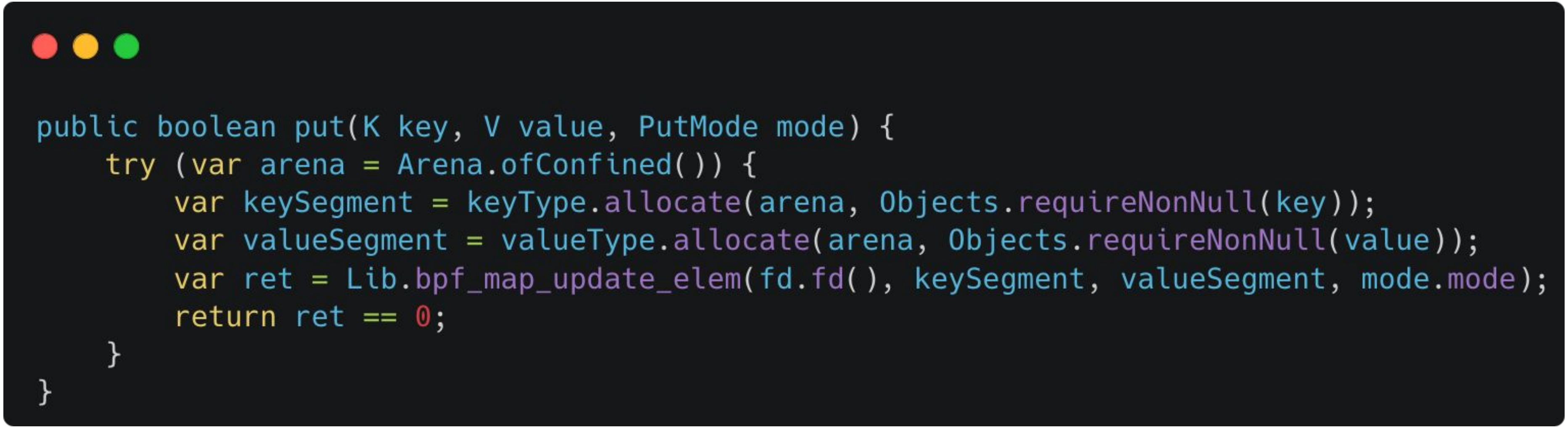
Any sufficiently advanced technology is indistinguishable from magic.

Clarke’s second law

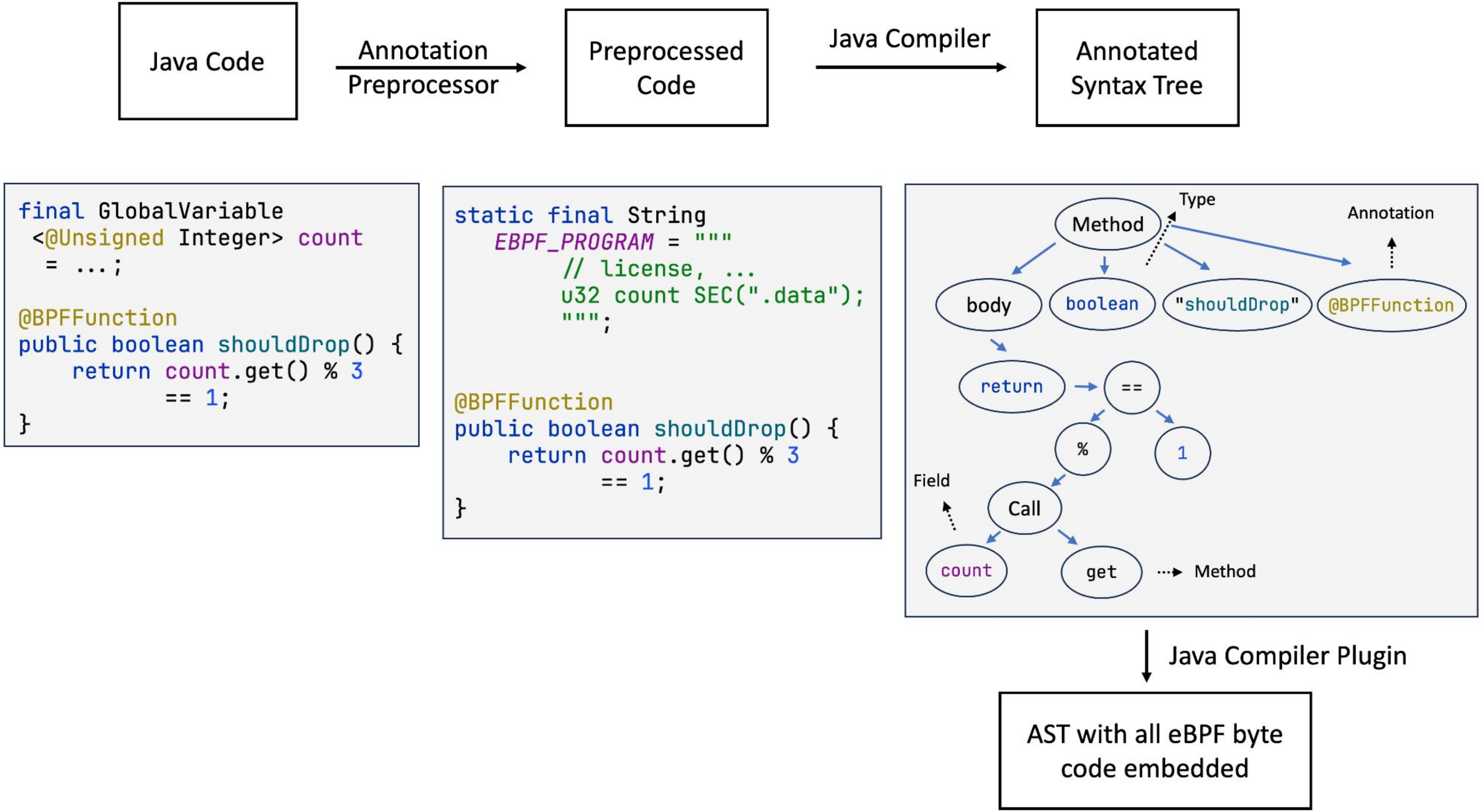
https://www.flickr.com/photos/itupictures/16636142906

History

Project Panama



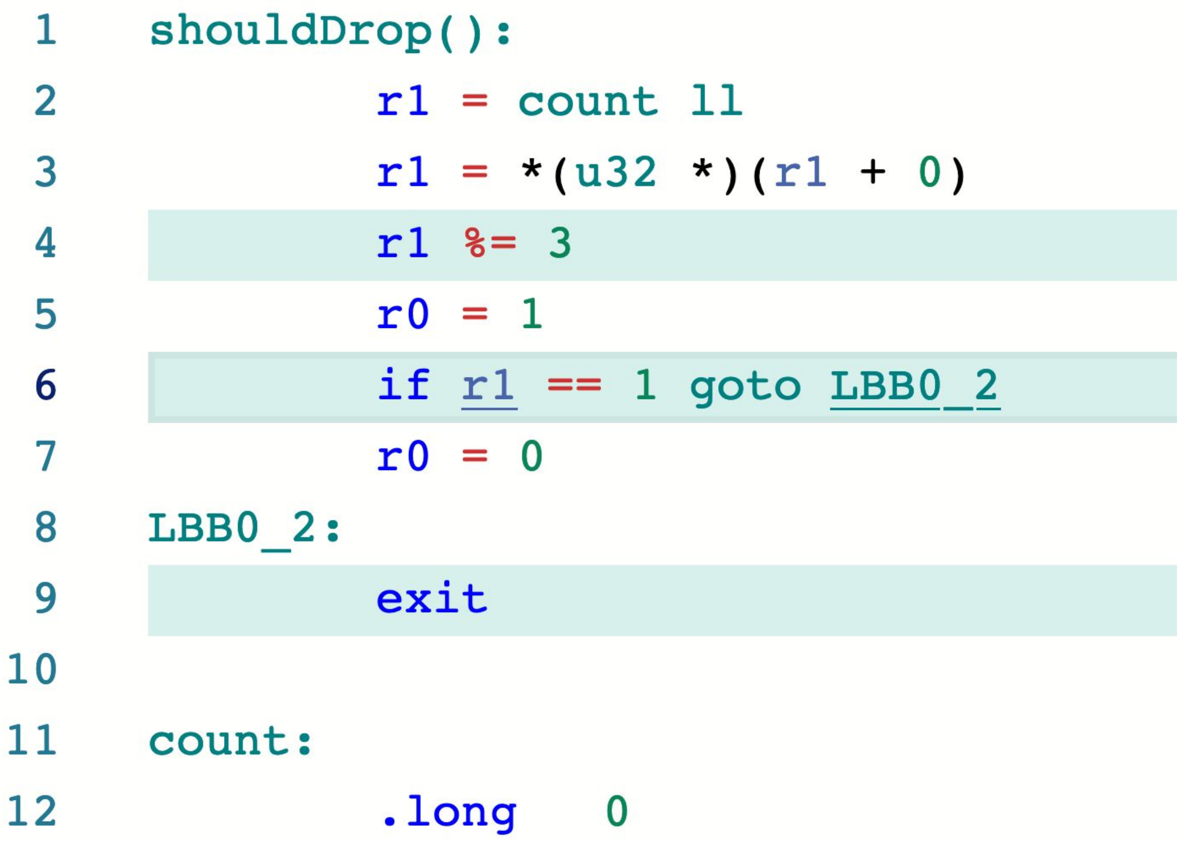
```java
public boolean put(K key, V value, PutMode mode) {
    try (var arena = Arena.ofConfined()) {
        var keySegment = keyType.allocate(arena, Objects.requireNonNull(key));
        var valueSegment = valueType.allocate(arena, Objects.requireNonNull(value));
        var ret = Lib.bpf_map_update_elem(fd.fd(), keySegment, valueSegment, mode.mode);
        return ret == 0;
    }
}
```



```
final GlobalVariable
 <@Unsigned Integer> count
  = ...;

@BPFFunction
public boolean shouldDrop() {
    return count.get() % 3
           == 1;
}
```

```
static final String
   EBPF_PROGRAM = """
        // license, ...
        u32 count SEC(".data");
        """;


@BPFFunction
public boolean shouldDrop() {
    return count.get() % 3
           == 1;
}
```

And for the compiler nerds



```
shouldDrop():
        r1 = count ll
        r1 = *(u32 *)(r1 + 0)
        r1 %= 3
        r0 = 1
        if r1 == 1 goto LBB0_2
        r0 = 0
LBB0_2:
        exit

count:
        .long   0
```

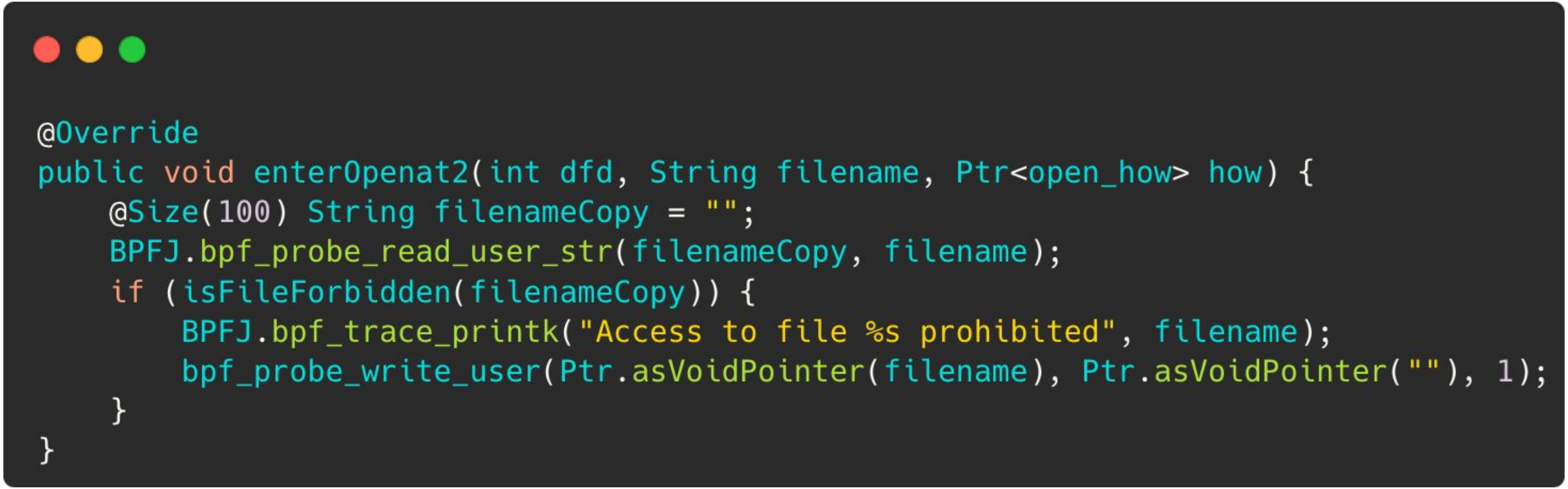
https://godbolt.org/z/9xoMzsc4b

Blog Posts

One every other week since January

JapMachine

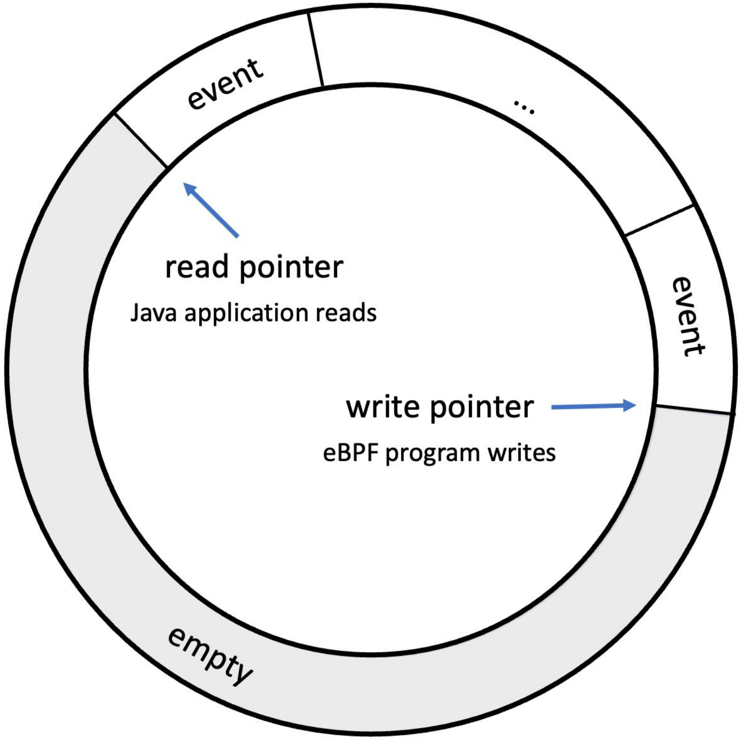
Hooks



```
@Override
public void enterOpenat2(int dfd, String filename, Ptr<open_how> how) {
    @Size(100) String filenameCopy = "";
    BPFJ.bpf_probe_read_user_str(filenameCopy, filename);
    if (isFileForbidden(filenameCopy)) {
        BPFJ.bpf_trace_printk("Access to file %s prohibited", filename);
        bpf_probe_write_user(Ptr.asVoidPointer(filename), Ptr.asVoidPointer(""), 1);
    }
}
```

https://embracethered.com/blog/posts/2021/offensive-bpf-libbpf-bpf_probe_write_user/

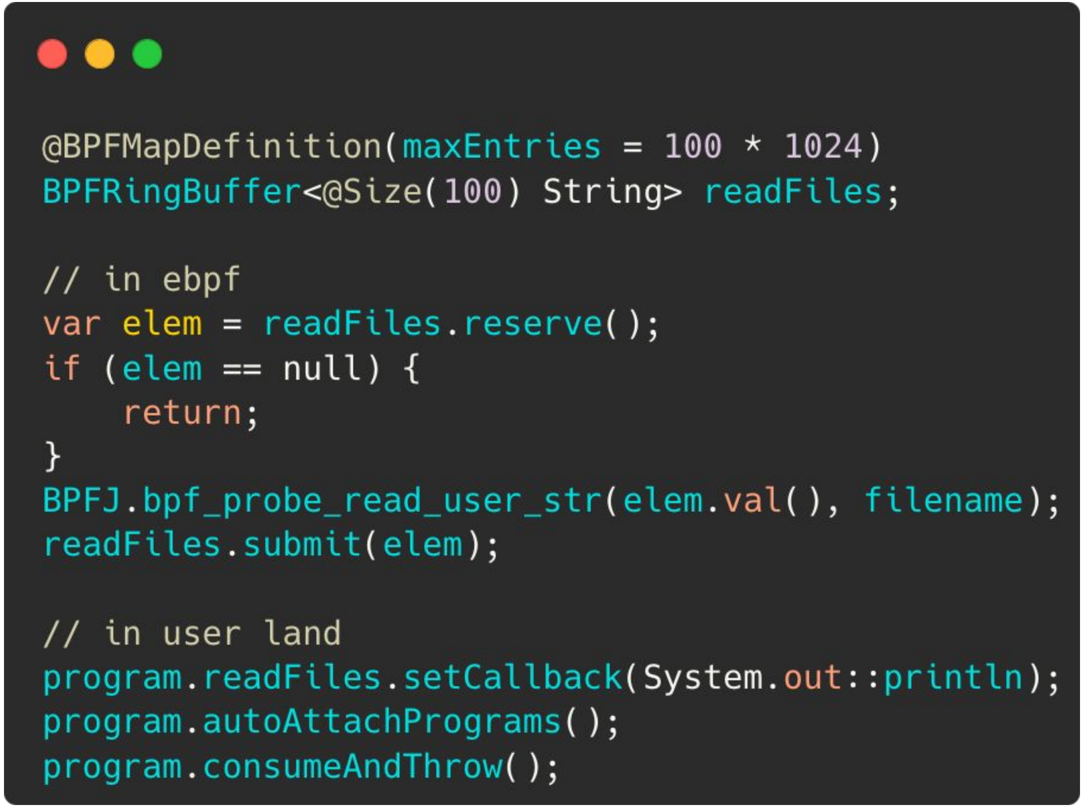
Ring Buffers



Ring Buffer for event

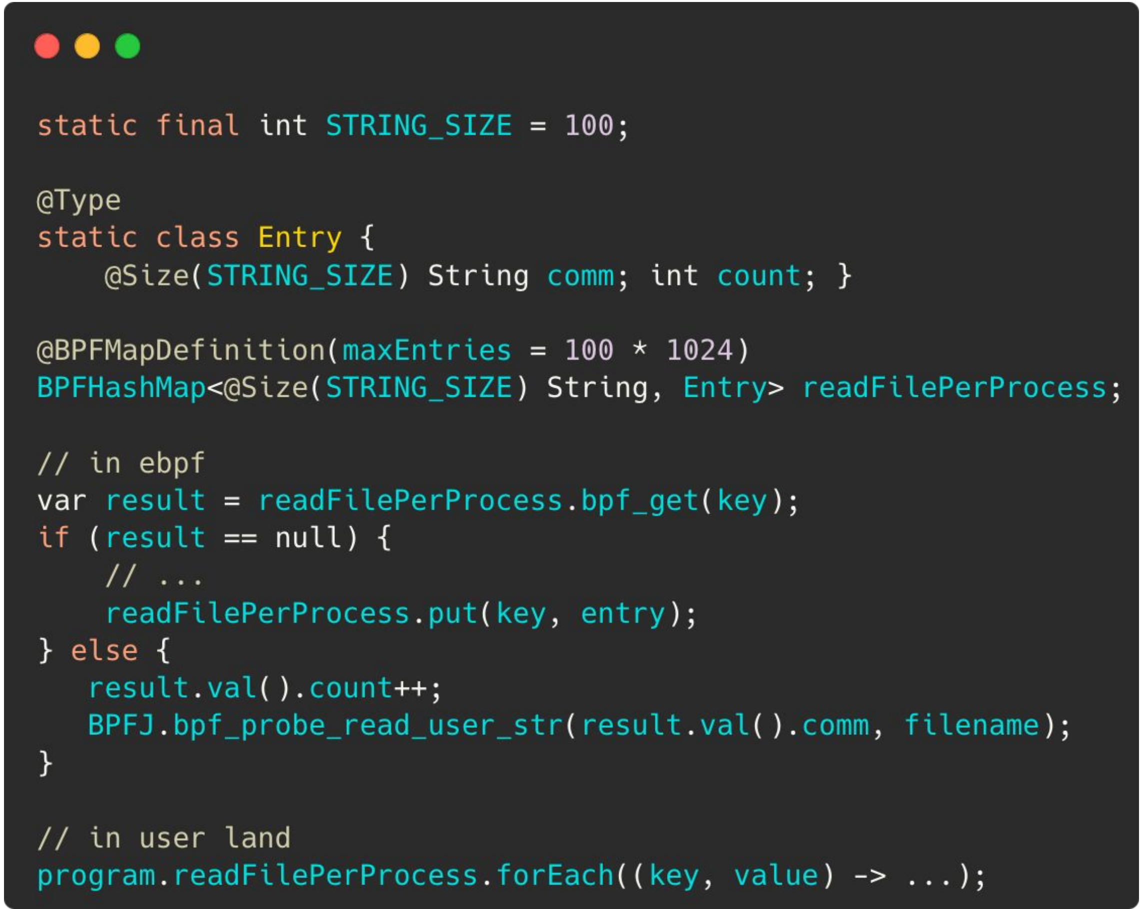
One for all CPUs

Ring Buffers



```
@BPFMapDefinition(maxEntries = 100 * 1024)
BPFRingBuffer<@Size(100) String> readFiles;

// in ebpf
var elem = readFiles.reserve();
if (elem == null) {
    return;
}
BPFJ.bpf_probe_read_user_str(elem.val(), filename);
readFiles.submit(elem);

// in user land
program.readFiles.setCallback(System.out::println);
program.autoAttachPrograms();
program.consumeAndThrow();
```

Maps



```
static final int STRING_SIZE = 100;

@Type
static class Entry {
    @Size(STRING_SIZE) String comm; int count; }

@BPFMapDefinition(maxEntries = 100 * 1024)
BPFHashMap<@Size(STRING_SIZE) String, Entry> readFilePerProcess;

// in ebpf
var result = readFilePerProcess.bpf_get(key);
if (result == null) {
    // ...
    readFilePerProcess.put(key, entry);
} else {
   result.val().count++;
   BPFJ.bpf_probe_read_user_str(result.val().comm, filename);
}

// in user land
program.readFilePerProcess.forEach((key, value) -> ...);
```

A glimpse into the future

Java as a first class language for eBPF

A glimpse into the future

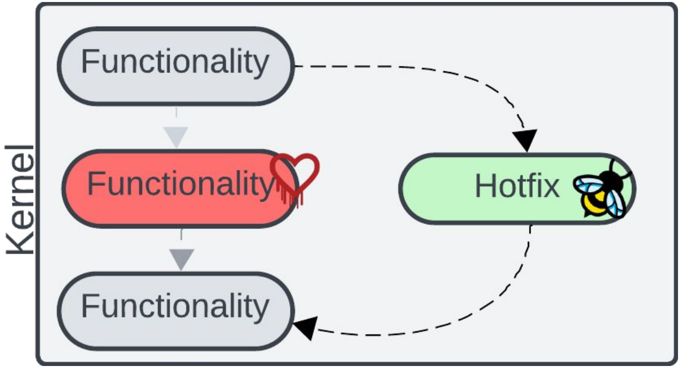
Towards a Micro Kernel

https://www.infoq.com/presentations/facebook-google-bpf-linux-kernel/

https://en.wikipedia.org/wiki/File:Pistachio_macro_whitebackground_NS.jpg

A glimpse into the future

Kernel Fixes Reimagined



A glimpse into the future

Sched Ext

The extensible sched_class

David Vernet
Kernel engineer

Kernel Recipes 2023 - sched_ext: pluggable scheduling in the Linux kernel

Final thoughts!

Thank you

Resources

hello eBPF

Thanks to

Dylan Reimerink

Sap Machine

Johannes Bechberger
@parttimen3rd
@parttimenerd@mastodon.social
me@mostlynerdless.de

Mohammed Aboullaite
@laytoun
mohammed@aboullaite.me