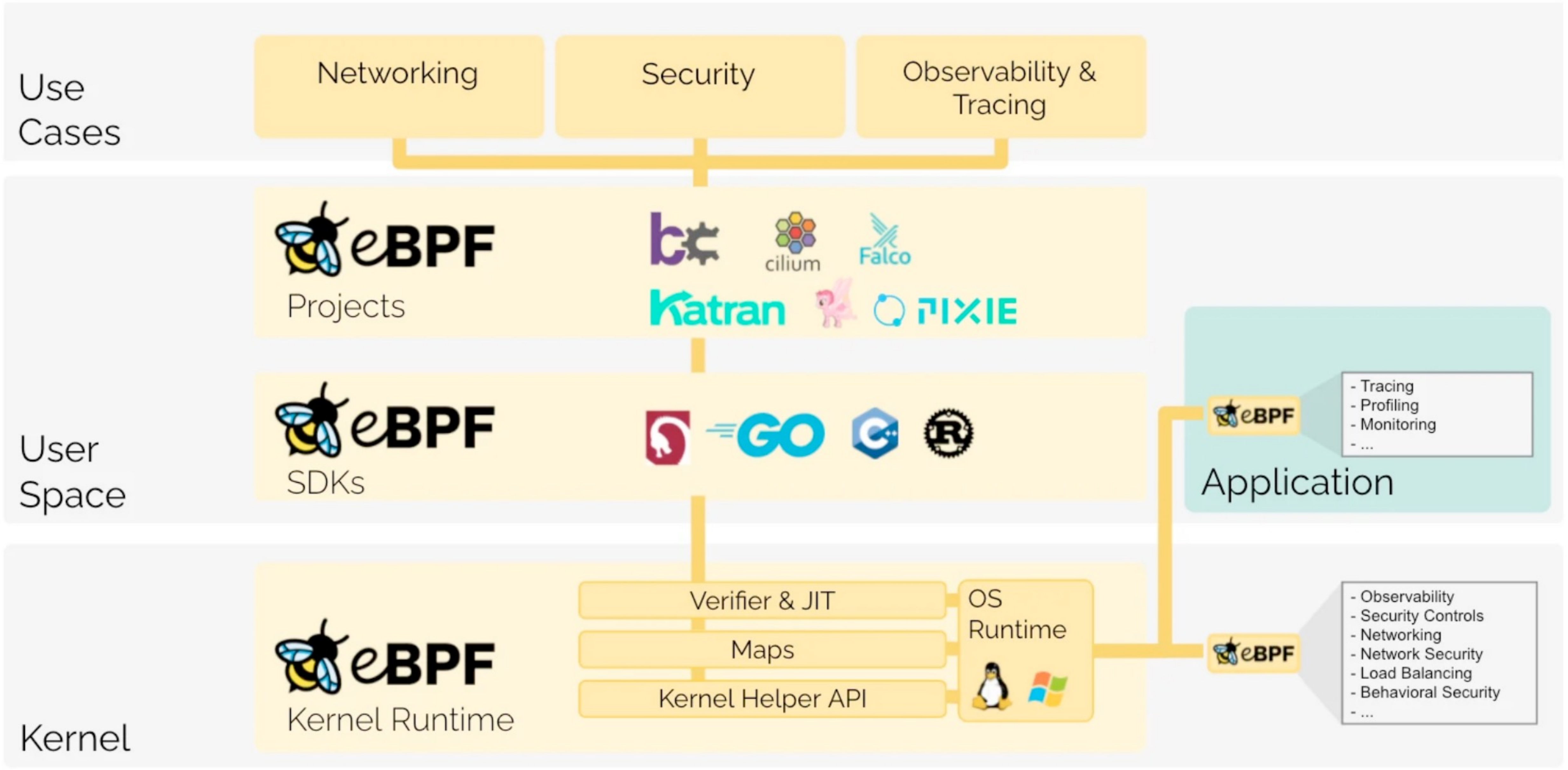
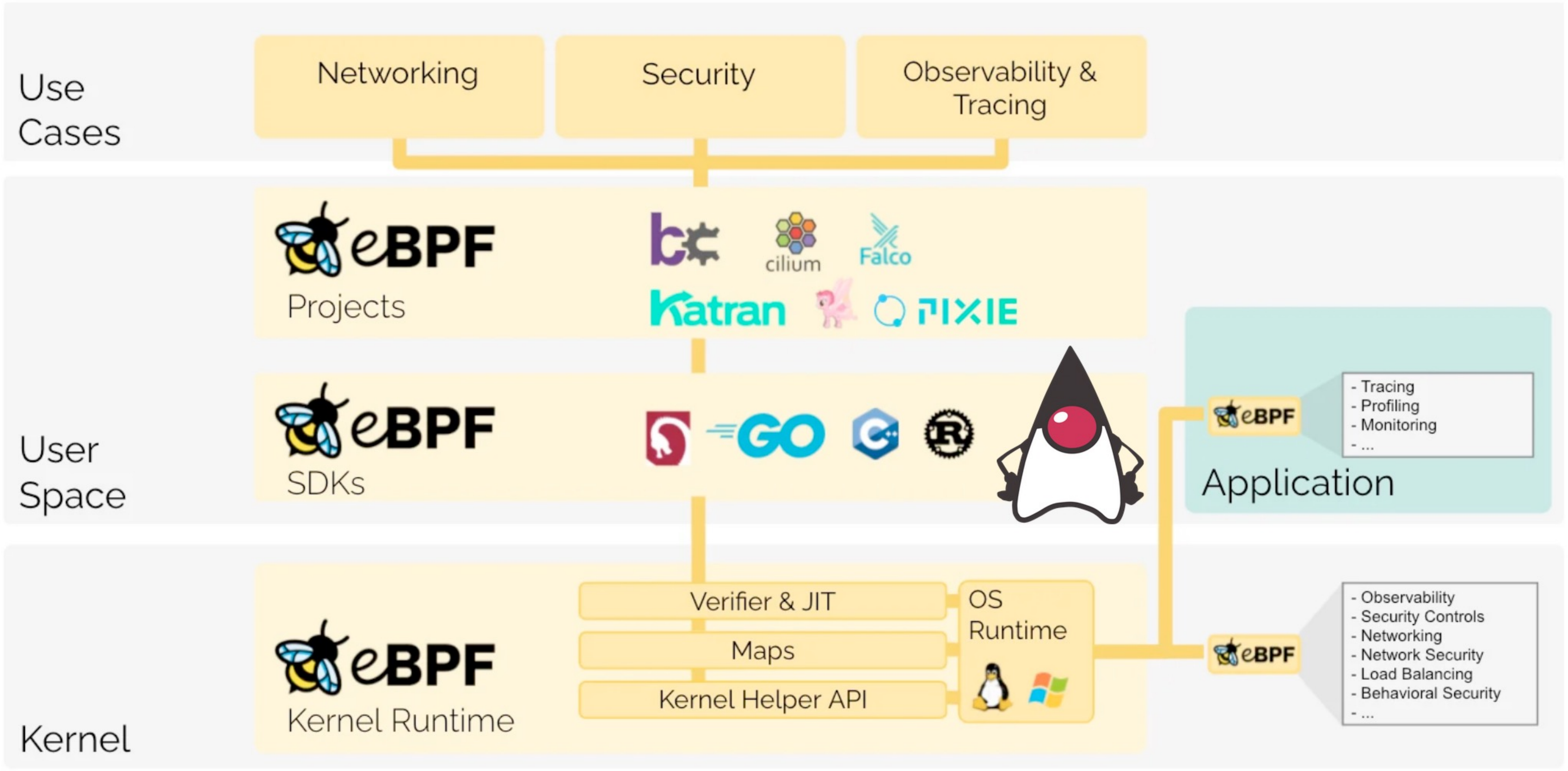
hello-ebpf: Writing eBPF programs directly in Java

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


Use Cases
Networking
Security
Observability & Tracing
User Space
eBPF Projects
cilium
Falco
Katran
PIXIE
eBPF SDKs
GO
Application
eBPF
- Tracing
- Profiling
- Monitoring
- ...
Kernel
eBPF Kernel Runtime
Verifier & JIT
Maps
Kernel Helper API
OS Runtime
eBPF
- Observability
- Security Controls
- Networking
- Network Security
- Load Balancing
- Behavioral Security
- ...



https://ebpf.io

But why?

You're crazy!

– Unnamed eBPF Developer

I thought it was a joke?

– Unnamed Linux Developer

Why not?

“

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

– Brendan Gregg

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

“

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

– Brendan Gregg

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

hello eBPF

still a prototype

How it all started

A version of libbcc-python in Java

```
def  perf_buffer_poll(self, timeout = -1):

        readers = (ct.c_void_p * len(self.perf_buffers))()

        for i, v in enumerate(self.perf_buffers.values()):
            readers[i] = v

        lib.perf_reader_poll(len(readers),       readers, timeout)
```

A version of libbcc-python in Java

```
void perf_buffer_poll(  int timeout     ) {
    try (var arena = Arena.ofConfined()) {
        var readers = arena.allocate(PanamaUtil.POINTER, perfBuffers.size());
        int i = 0;
        for (var v : perfBuffers.values()) {
            readers.setAtIndex(POINTER, i++, v);
        }
        Lib.perf_reader_poll(perfBuffers.size(), readers, timeout);
    }
}
```

A version of libbcc-python in Java

```
public class HelloWorld {
    public static void main(String[] args) {
        try (BPF b = BPF.builder("""
        int hello(void *ctx) {
           bpf_trace_printk("Hello, World!");
           return 0;
        }
        """).build()) {
            var syscall = b.get_syscall_fnname("execve");
            b.attach_kprobe(syscall, "hello");
            b.trace_print();
        }
    }
}
```

Compile and load

Attach

Start print-loop for trace log

But libbcc has problems

and writing Pythonic code in Java has them too

So I rewrote everything

and sprinkled in some magic

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

    @Override
```

Kernel

```java
    public void enterOpenat2(int dfd, String filename,
                                         Ptr<open_how> how) {
        bpf_trace_printk("Open %s", filename);
    }
```

Userland/Java

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

License of the eBPF Program

Program name

Interface with Annotations

```
@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("Open %s", filename);
    }

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

Kernel

Userland/Java

```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("Open %s", filename);
    }

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

Kernel

Userland/Java

Print to trace log

```
@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("Open %s", filename);
    }

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

Kernel

Userland/Java

Load into Kernel

Attach to fenter, kprobes, …

Start print-loop for trace log

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

    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;
    }

    // ...
}
```

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

    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;
    }

    // ...
}
```

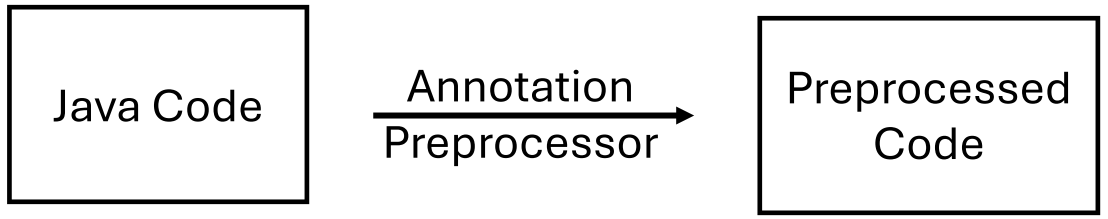
```java
@BPF(license = "GPL")
public abstract class XDPDropEveryThirdPacket
extends BPFProgram implements XDPHook {

    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;
    }

    // ...
}
```

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

    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;
    }

    // ...
}
```

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

    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;
    }

    // ...
}
```

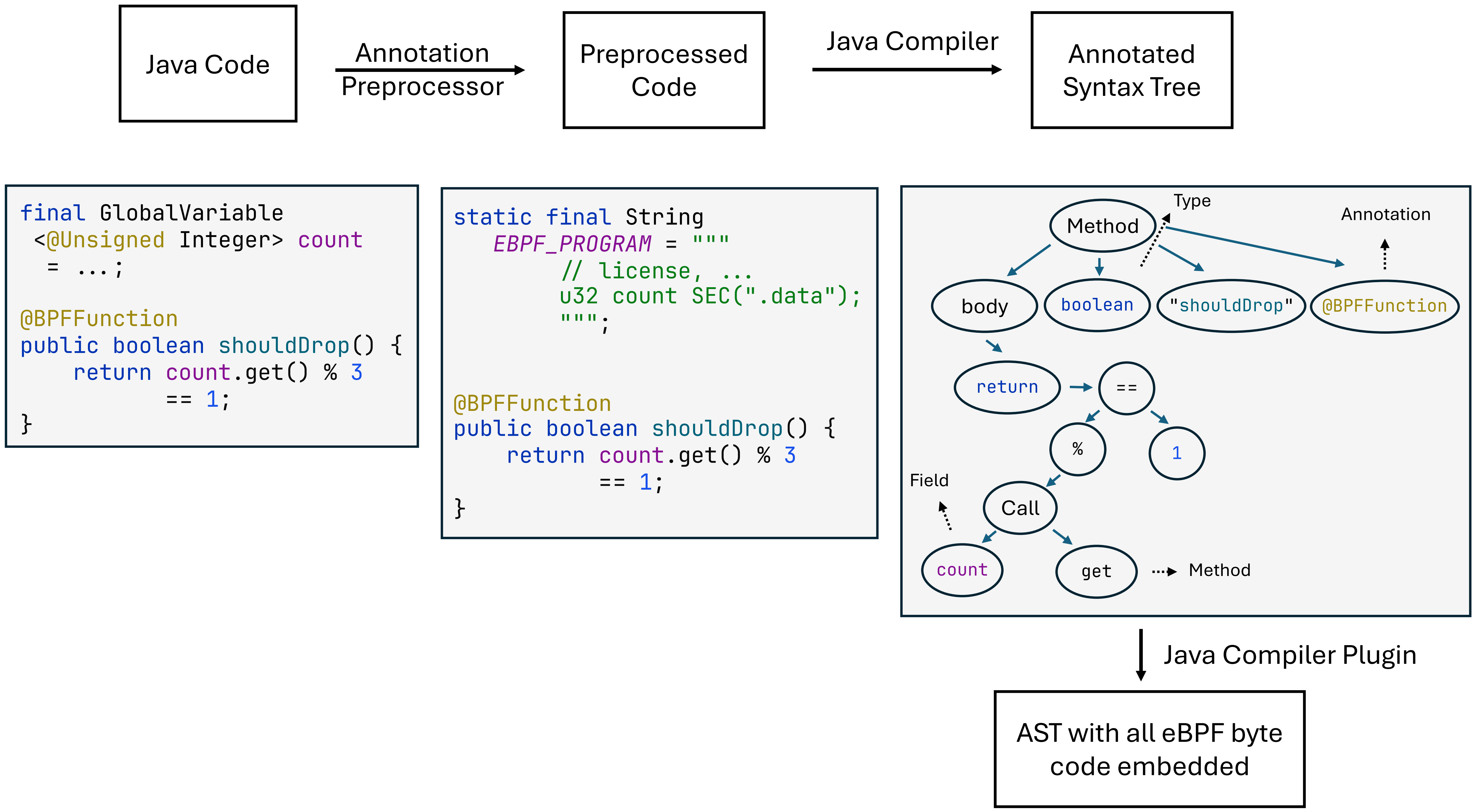
Java Code

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

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



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

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

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


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



```
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;
}
```

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

    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;
    }

    // ...
}
```

```c
u32 count SEC(".data");

bool shouldDrop() {
    return ((count) % 3) == 1;
}

SEC("xdp") enum xdp_action xdpHandlePacket(struct xdp_md *ctx) {
    count = ((count) + 1);
    return shouldDrop() ? XDP_DROP : XDP_PASS;
}
```

How to model C…

Structs

```
@Type
struct Event {
    @Unsigned int pid;
    @Size(256) String f;
}

@BPFFunction
int access(       Event event) {
    event.pid = 5;
    return event.pid;
}
```

Structs

```
struct Event {
    u32 pid;
    u8 filename[256];
};

s32 access(struct Event event) {
    event.pid = 5;
    return event.pid;
}
```

Unions

```
@Type
static class SampleUnion extends Union {
    @Unsigned
    int ipv4;
    long count;
}
```

Pointers Ptr<T>

```
@BPFFunction
public int refAndDeref() {
    int value = 3;
    Ptr<Integer> ptr = Ptr.of(value);
    return ptr == Ptr.ofNull() ? 1 : 0;
}
```

Pointers   Ptr<T>

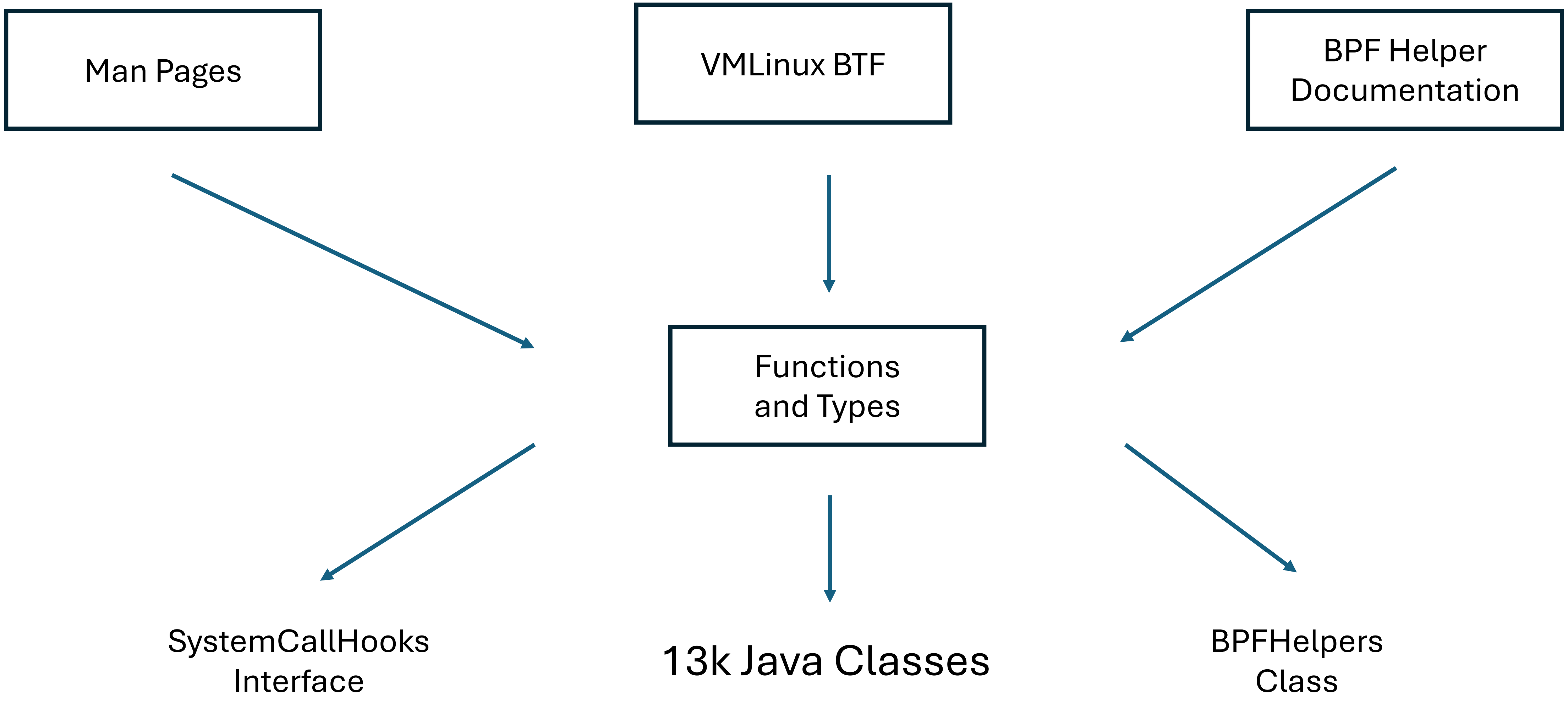
```
s32 refAndDeref() {
    s32 value = 3;
    s32*          ptr = &value;
    return ptr == ((void*)0)   ? 1 : 0;
}
```

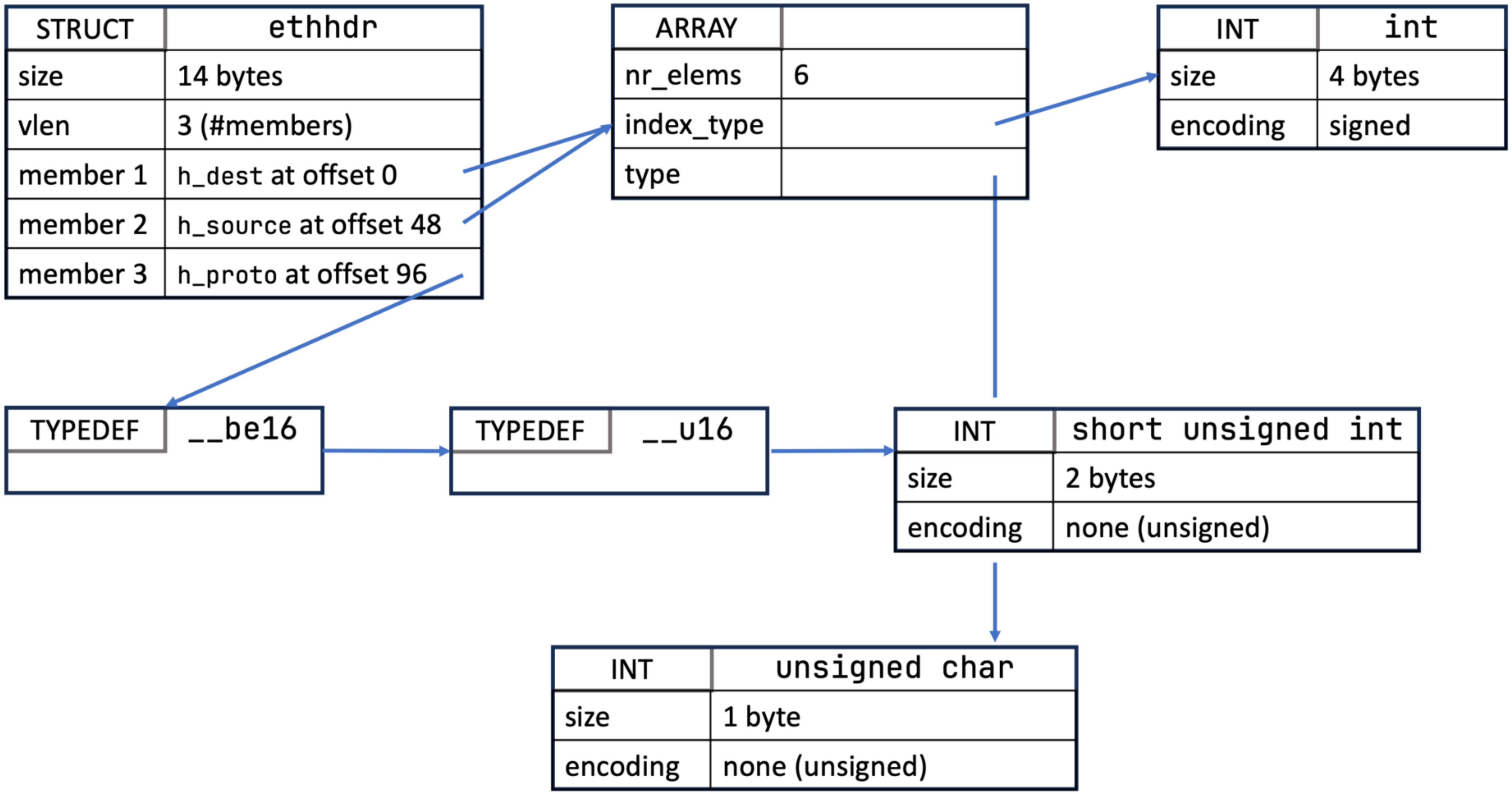
```java
public class Ptr<T> {

    @BuiltinBPFFunction("(*($this))")
    public T val() {}

    @BuiltinBPFFunction("&($arg1)")
    public static <T> Ptr<T> of(@Nullable T value){}

    @BuiltinBPFFunction("((void*)0)")
    public static Ptr<?> ofNull() {}

    @BuiltinBPFFunction("(($T1*)$this)")
    public <S> Ptr<S> cast() {}

    // ...
}
```

Maps

```
@BPFMapDefinition(maxEntries = 100 * 1024)
BPFHashMap<@Size(STRING_SIZE) String, Entry> map;
// in eBPF
String key = ...;
map.bpf_get(key); map.put(key, entry);
// in Java
program.map.get(key)
program.map.put(key, value)
```

VMLinux


Man Pages
VMLinux BTF
BPF Helper Documentation
Functions and Types
SystemCallHooks Interface
13k Java Classes
BPFHelpers Class


STRUCT
ethhdr
size
14 bytes
vlen
3 (#members)
member 1
h_dest at offset 0
member 2
h_source at offset 48
member 3
h_proto at offset 96
ARRAY
nr_elems
6
index_type
type
INT
int
size
4 bytes
encoding
signed
TYPEDEF
__be16
TYPEDEF
__u16
INT
short unsigned int
size
2 bytes
encoding
none (unsigned)
INT
unsigned char
size
1 byte
encoding
none (unsigned)

Is based on

```
struct ethhdr {
           unsigned  char h_dest[ETH_ALEN];

           unsigned  char h_source[ETH_ALEN];

           __be16                                  h_proto;
} __attribute__((packed));
```

Is turned into

```
@Type(
        noCCodeGeneration = true,
        cType = "struct ethhdr"
)
@NotUsableInJava // mark as eBPF only
public static class ethhdr extends Struct {
    public @Unsigned char @Size(6) [] h_dest;

    public @Unsigned char @Size(6) [] h_source;

    public @Unsigned @OriginalName("__be16") short h_proto;
}
```

But there are also classes like

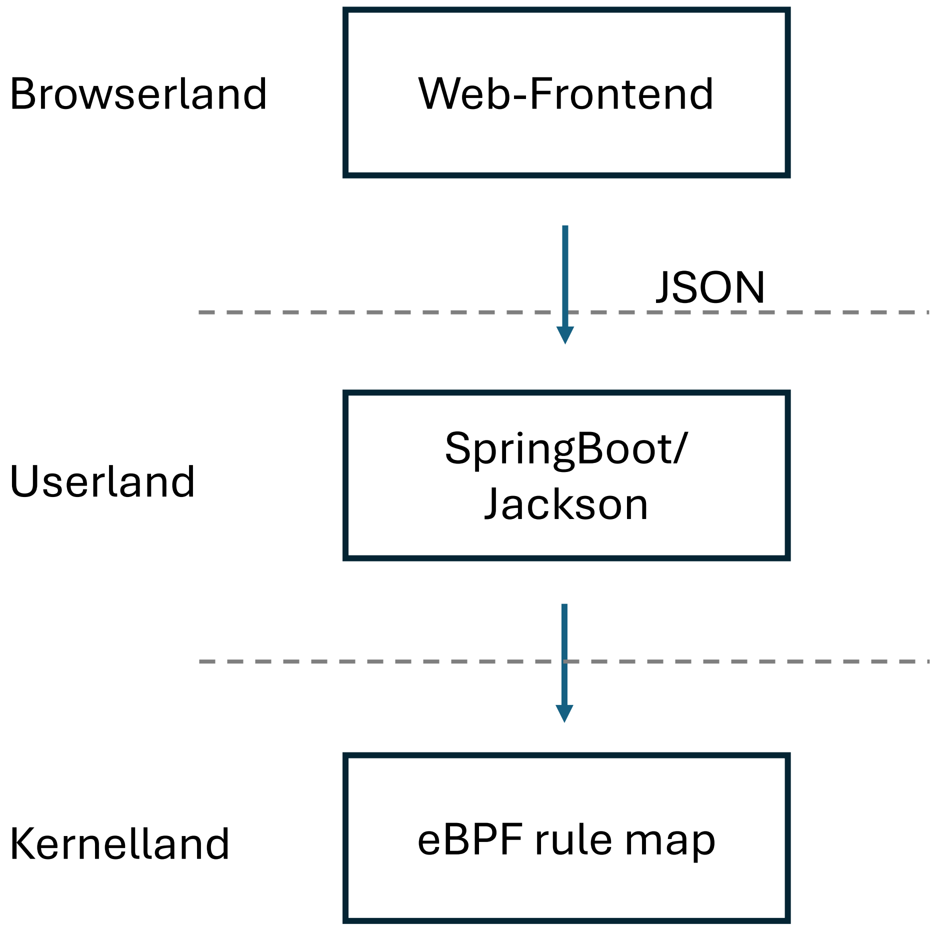
```java
@Type(
    noCCodeGeneration = true, cType = "struct xdp_page_head")
@NotUsableInJava
public static class xdp_page_head extends Struct {
    public xdp_buff orig_ctx;
    public xdp_buff ctx;
    @InlineUnion(25132)
    public AnonDefinitions.@InlineUnion(25132)
anon_member_of_anon_member_of_xdp_page_head anon2$0;
    // ...

    public xdp_page_head() {
    }
}
```

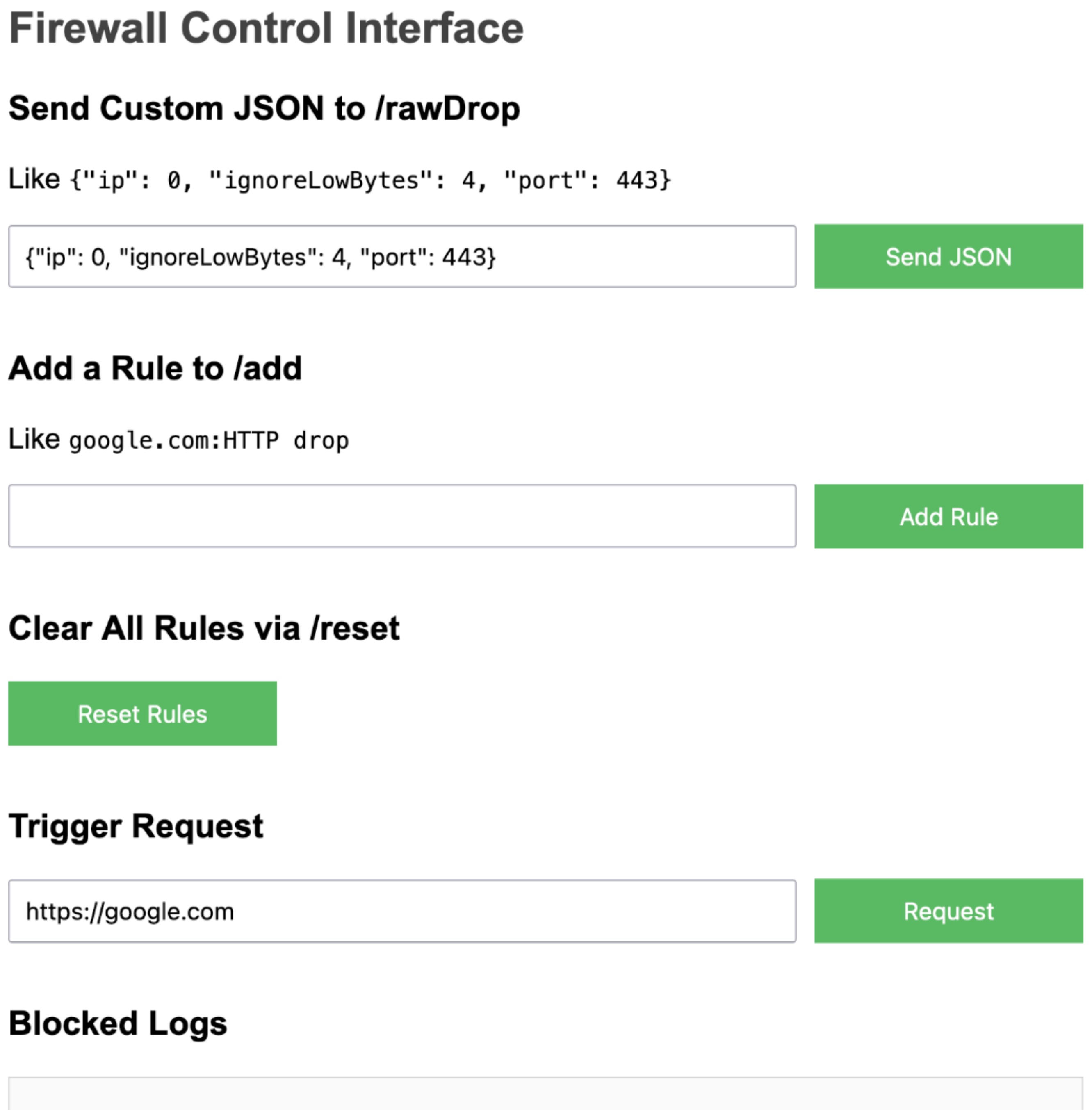
To write code like

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

What can we do?

Firewall





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`

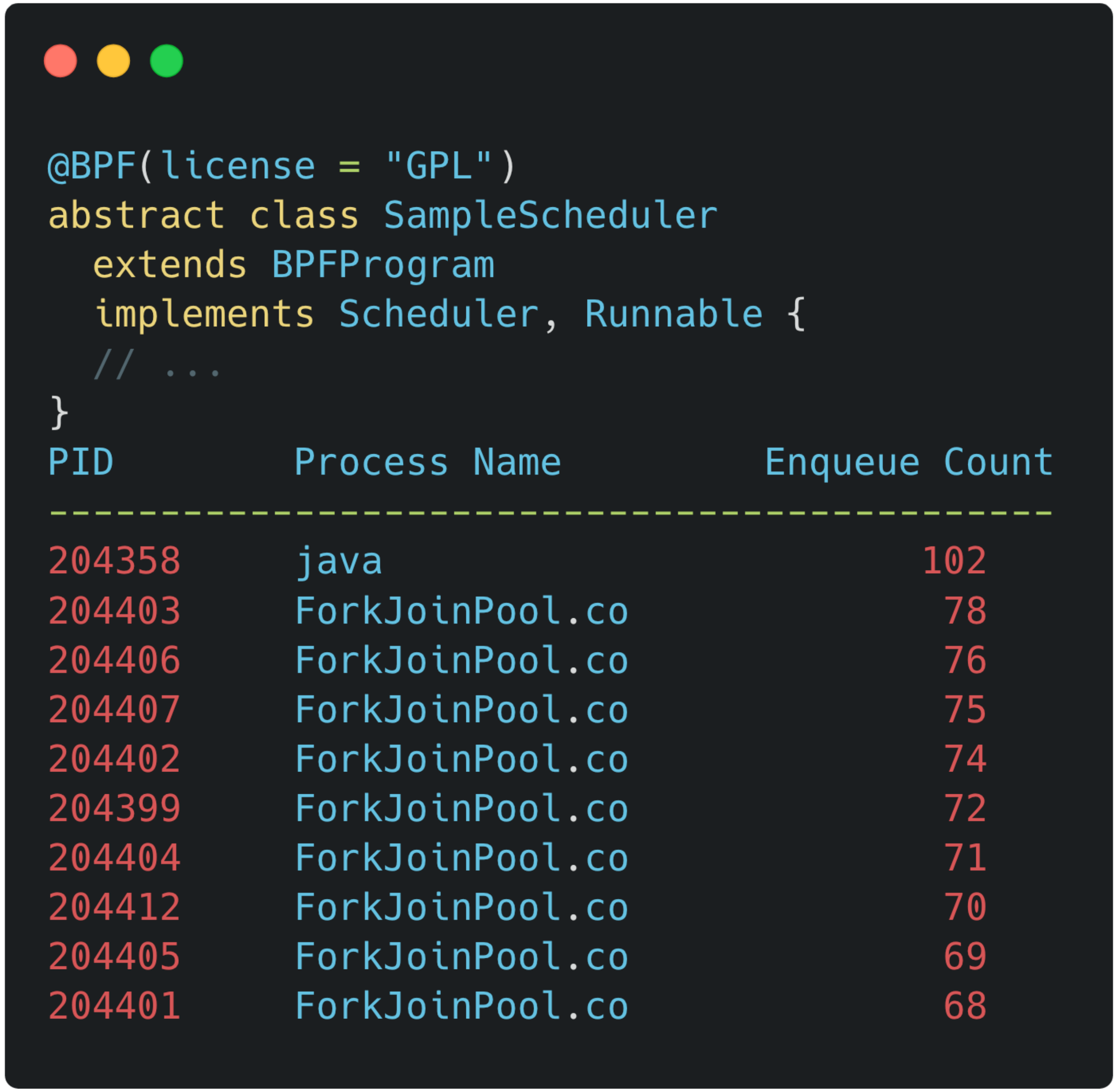
Clear All Rules via /reset

Trigger Request

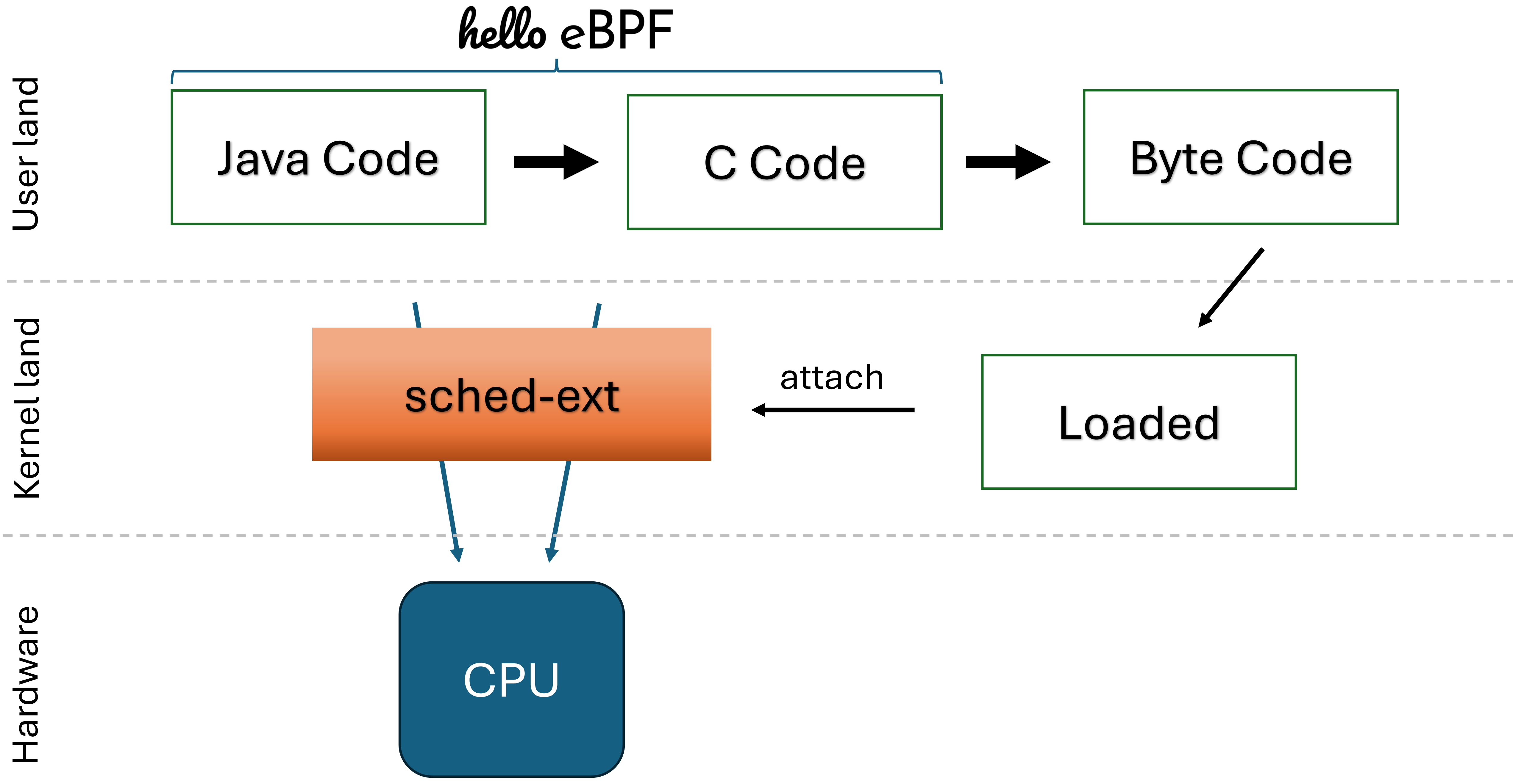
https://google.com

Blocked Logs

Schedulers



```
@BPF(license = "GPL")
abstract class SampleScheduler
  extends BPFProgram
  implements Scheduler, Runnable {
  // ...
}
PID          Process Name          Enqueue Count
------------------------------------------------
204358       java                           102
204403       ForkJoinPool.co                 78
204406       ForkJoinPool.co                 76
204407       ForkJoinPool.co                 75
204402       ForkJoinPool.co                 74
204399       ForkJoinPool.co                 72
204404       ForkJoinPool.co                 71
204412       ForkJoinPool.co                 70
204405       ForkJoinPool.co                 69
204401       ForkJoinPool.co                 68
```


hello eBPF
User land
Java Code
C Code
Byte Code
Kernel land
sched-ext
attach
Loaded
Hardware
CPU

Fin.

A blog post
every two weeks

hello eBPF

Thanks to

Dylan Reimerink

Johannes Bechberger
mostlynerdless.de
OpenJDK Developer, SAP