



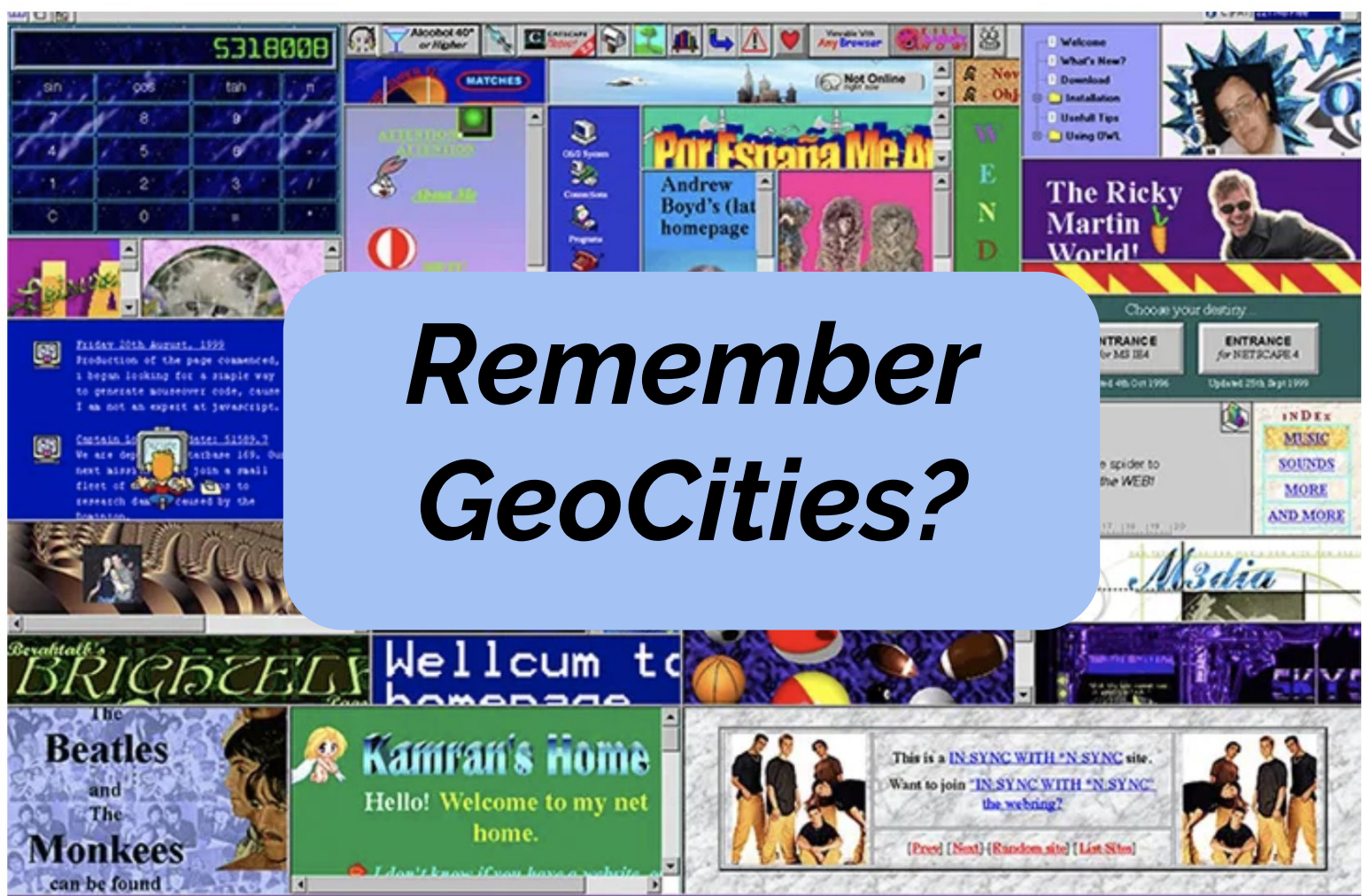
eBPF

Rethinking the Linux kernel

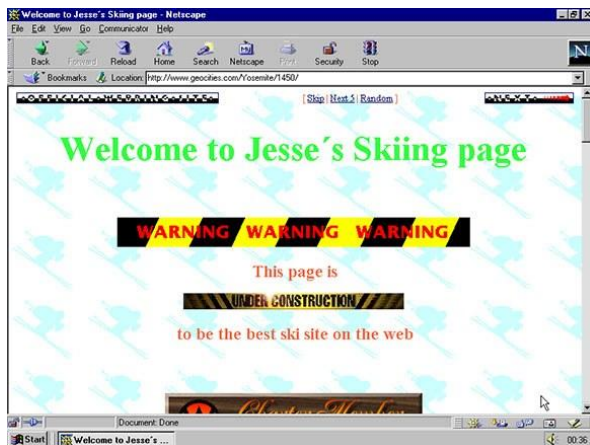
Thomas Graf

Cilium Project, Co-Founder & CTO, Isovalent





What enabled this evolution?



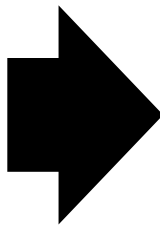
Markup Only (HTML)

Network Working Group
Request for Comments: 1866
Category: Standards Track

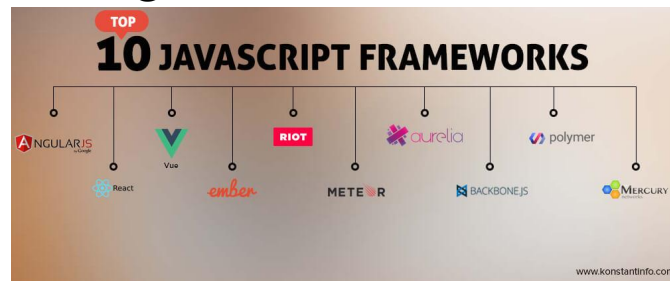
T. Berners-Lee
MIT/W3C
D. Connolly
November 1995

Hypertext Markup Language - 2.0

Status of this Memo



Programmable Platform



Programmability Essentials



Safety

Untrusted code runs in the browser of the user.

→ *Sandboxing*



Continuous Delivery

Allow evolution of logic without requiring to constantly ship new browser versions.

→ *Deploy anytime with seamless upgrades*

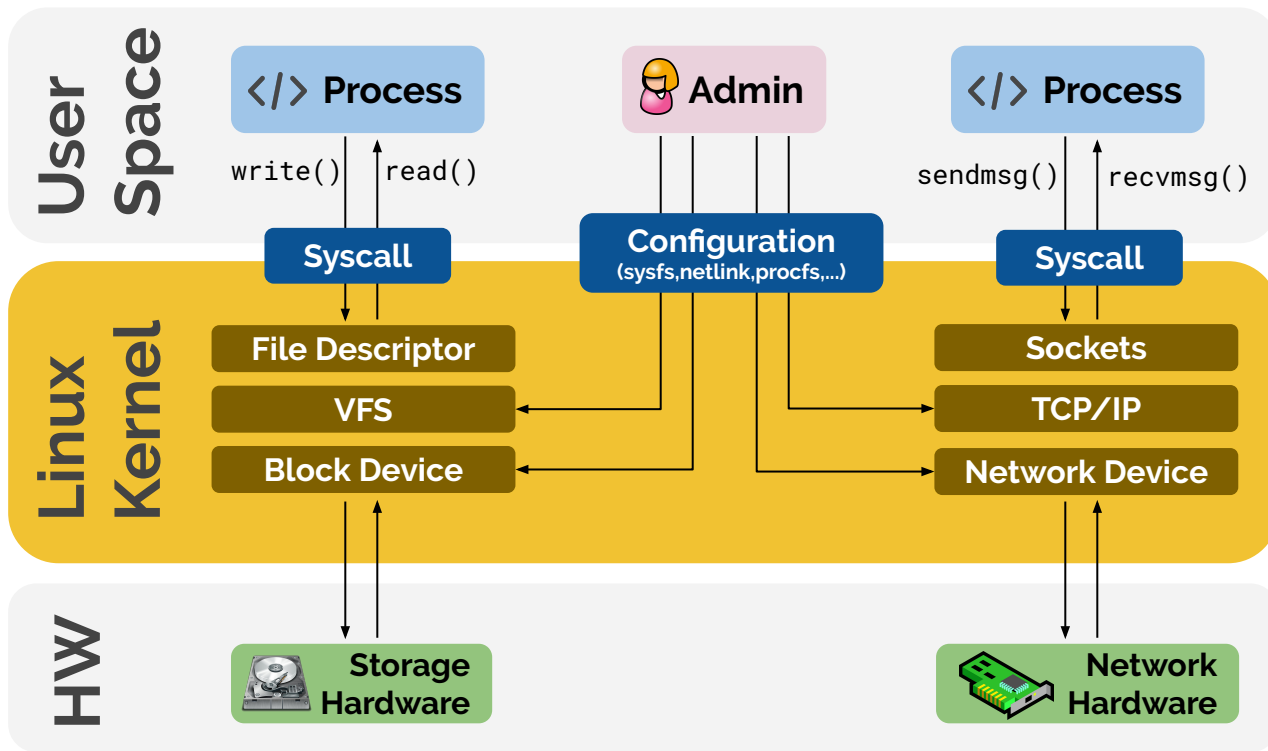


Performance

Programmability must be provided with minimal overhead.

→ *Native Execution (JIT compiler)*

Kernel Architecture



Kernel Development 101

Option 1

Native Support

- Change kernel source code
- Expose configuration API
- Wait 5 years for your users to upgrade

Cons:



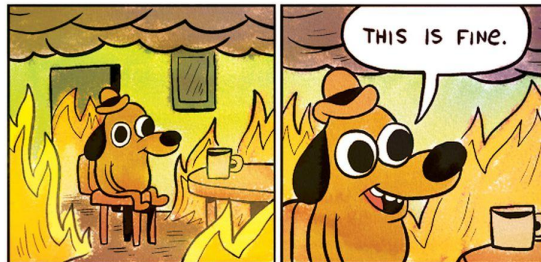
Option 2

Kernel Module

- Write kernel module
- Every kernel release will break it

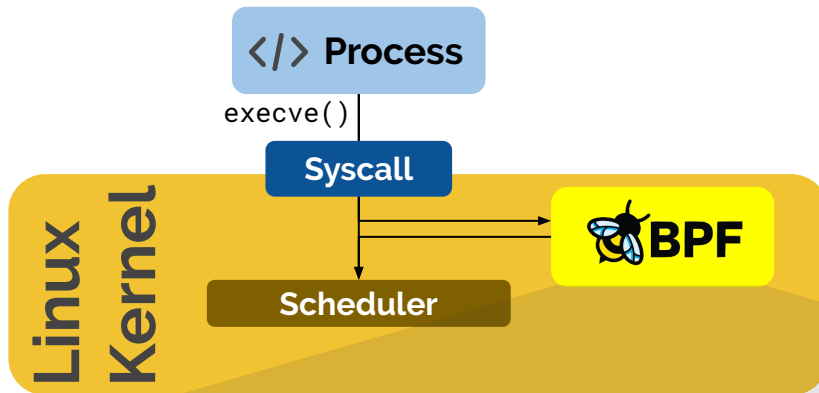
Cons:

- You likely need to ship a different module for each kernel version
- Might crash your kernel



*How about we add
JavaScript-like capabilities
to the Linux Kernel?*



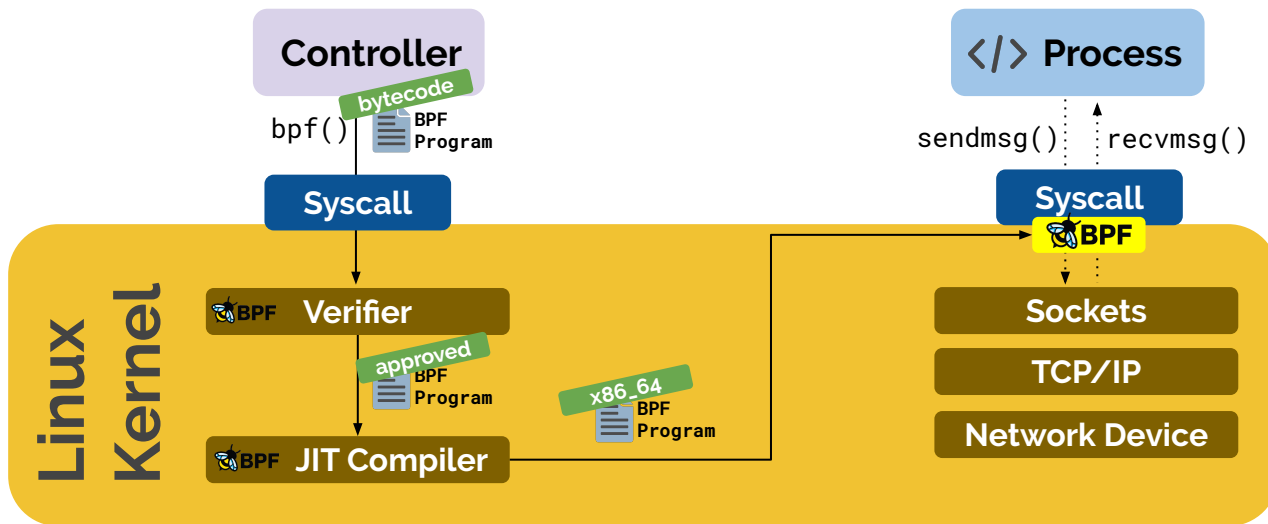


```
int syscall__ret_execve(struct pt_regs *ctx)
{
    struct comm_event event = {
        .pid = bpf_get_current_pid_tgid() >> 32,
        .type = TYPE_RETURN,
    };

    bpf_get_current_comm(&event.comm, sizeof(event.comm));
    comm_events.perf_submit(ctx, &event, sizeof(event));

    return 0;
}
```

eBPF Runtime



Safety & Security

The verifier will reject any unsafe program and provides a sandbox.



Continuous Delivery

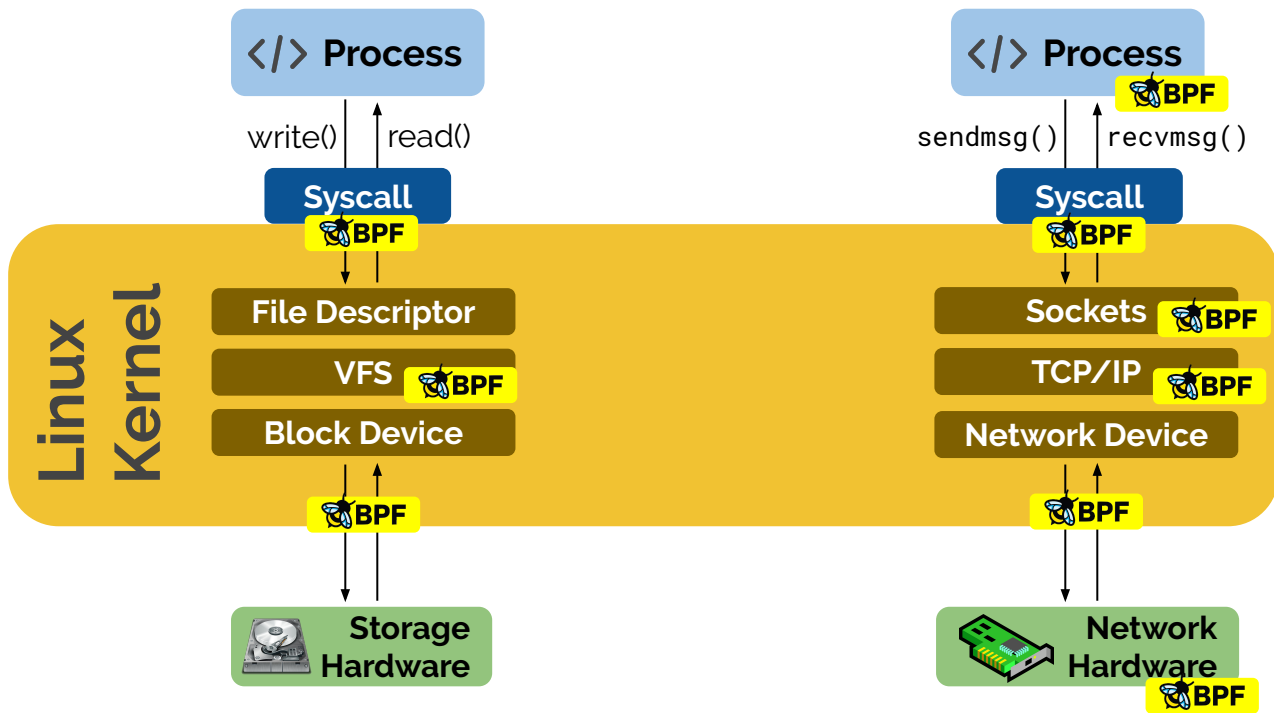
Programs can be exchanged without disrupting workloads.



Performance

The JIT compiler ensures native execution performance.

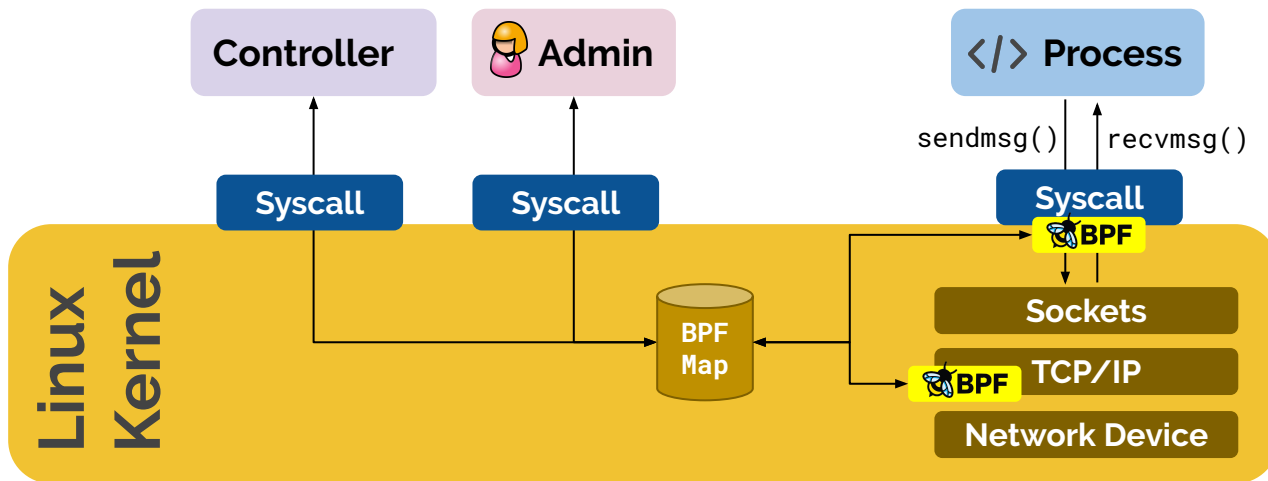
eBPF Hooks



Where can you hook? kernel functions (kprobes), userspace functions (uprobes), system calls, fentry/fexit, tracepoints, network devices (tc/xdp), network routes, TCP congestion algorithms, sockets (data level)



eBPF Maps



Map Types:

- Hash tables, Arrays
- LRU (Least Recently Used)
- Ring Buffer
- Stack Trace
- LPM (Longest Prefix match)

What are Maps used for?

- Program state
- Program configuration
- Share data between programs
- Share state, metrics, and statistics with user space



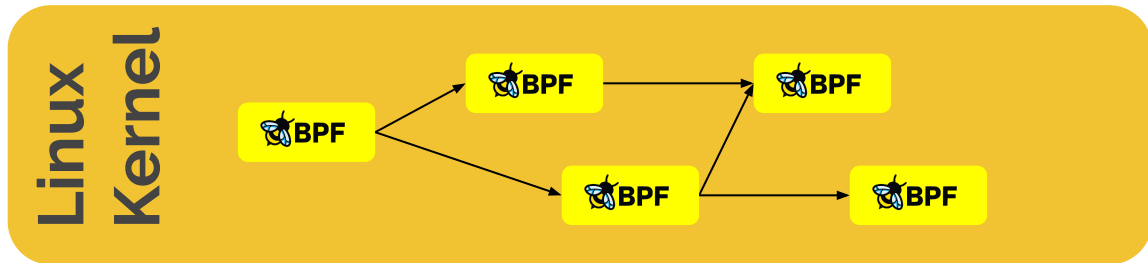
eBPF Helpers



What helpers exist?

- Random numbers
- Get current time
- Map access
- Get process/cgroup context
- Manipulate network packets and forwarding
- Access socket data
- Perform tail call
- Access process stack
- Access syscall arguments
- ...

eBPF Tail and Function Calls



What are Tail Calls used for?

- Chain programs together
- Split programs into independent logical components
- Make BPF programs composable

What are Functions Calls used for?

- Reuse functionality inside of a program
- Reduce program size (avoid inlining)

eBPF Community

287 contributors:

(Jan 2016 to Jan 2020)

- 466 Daniel Borkmann (Cilium; maintainer)
- 290 Andrii Nakryiko (Facebook)
- 279 Alexei Starovoitov (Facebook; maintainer)
- 217 Jakub Kicinski (Facebook)
- 173 Yonghong Song (Facebook)
- 168 Martin KaFai Lau (Facebook)
- 159 Stanislav Fomichev (Google)
- 148 Quentin Monnet (Cilium)
- 148 John Fastabend (Cilium)
- 118 Jesper Dangaard Brouer (Red Hat)
- [...]



eBPF Projects



Katran

High-performance L4
Loadbalancer

[facebookincubator/katran](https://facebookincubator.com/katran)



Cilium

Networking, security and
load-balancing for k8s

[cilium/cilium](https://cilium.io)



bcc, bpftrace

Performance
troubleshooting &
profiling

[iovisor/bcc](https://github.com/iovisor/bcc)



Android & Security

kernel runtime security
instrumentation (**KRSI**),
[Android BPF loader](#),
[eBPF traffic monitor](#)



Traffic Optimization

DDoS mitigation, QoS,
traffic optimization,
[load balancer](#)

[cloudflare/bpftools](https://cloudflare.com/bpftools)



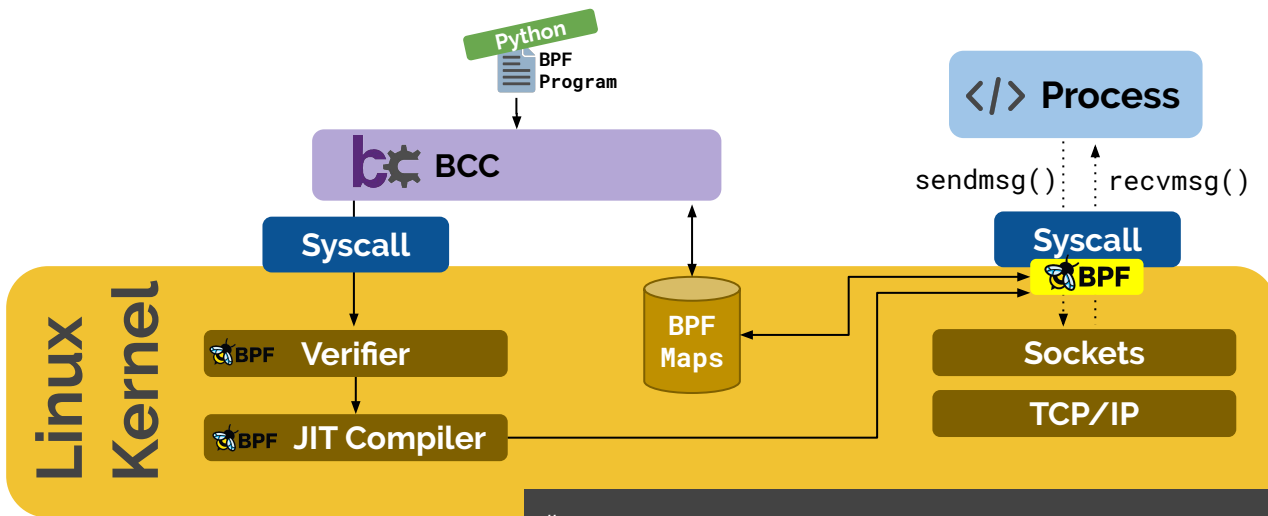
Falco

Container runtime
security, behavior
analysis

[falcosecurity/falco](https://falcosecurity.com/falco)



Tracing & Profiling with eBPF



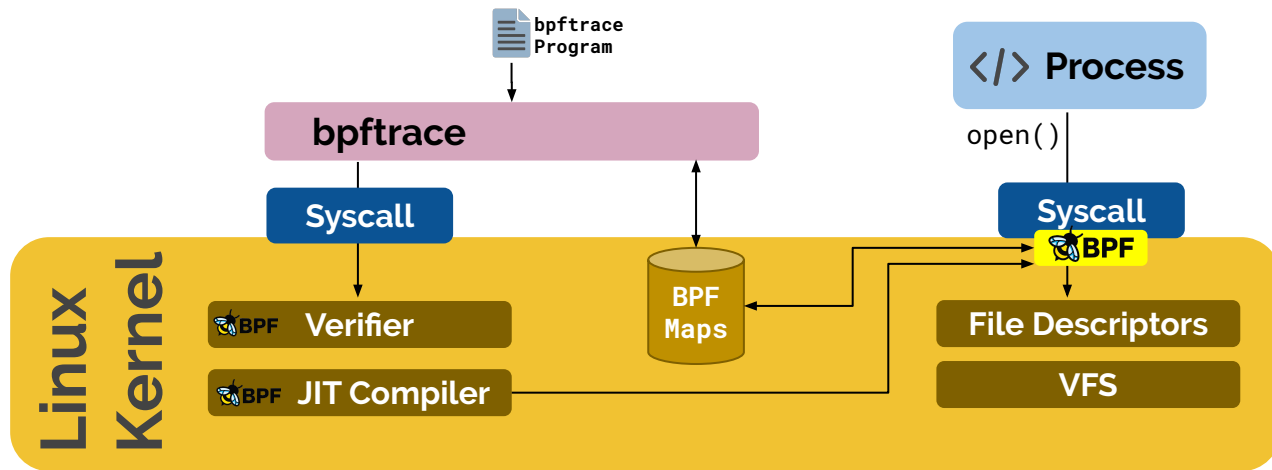
BCC:

github.com/iovisor/bcc

```
# tcpdump
Tracing... Output every 1 secs. Hit Ctrl-C to end
<screen clears>
19:46:24 loadavg: 1.86 2.67 2.91 3/362 16681
```

PID	COMM	LADDR	RADDR	RX_KB	TX_KB
16648	16648	100.66.3.172:22	100.127.69.165:6684	1	0
16647	sshd	100.66.3.172:22	100.127.69.165:6684	0	2149
14374	sshd	100.66.3.172:22	100.127.69.165:25219	0	0
14458	sshd	100.66.3.172:22	100.127.69.165:7165	0	0

bpfttrace - DTrace for Linux



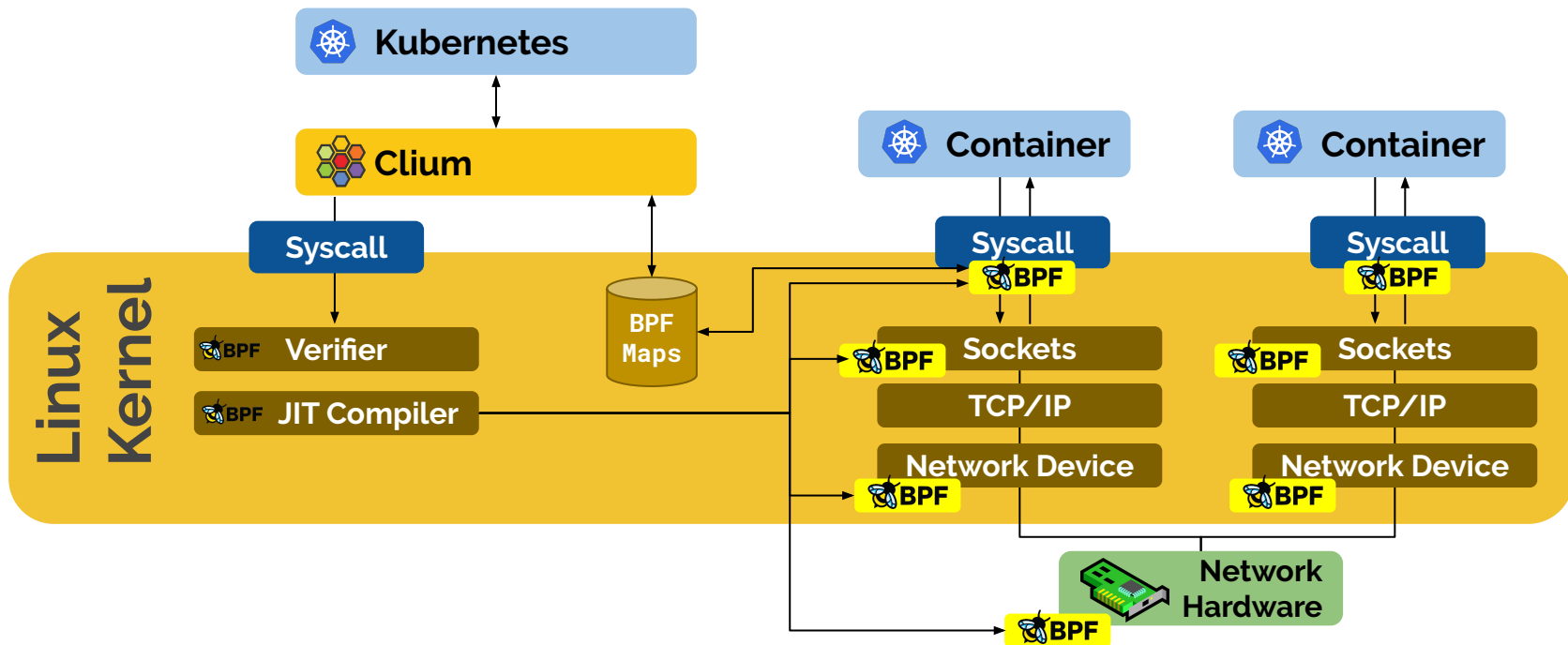
bpfttrace:

github.com/iovisor/bpfttrace

```
# bpfttrace -e 'kprobe:do_sys_open { printf("%s: %s\n", comm, str(arg1)) }'
Attaching 1 probe...
git: .git/objects/da
git: .git/objects/pack
git: /etc/localtime
systemd-journal: /var/log/journal/72d0774c88dc4943ae3d34ac356125dd
DNS Res~ver #15: /etc/hosts
^C
```



Networking, load-balancing and security for Kubernetes





Container Networking

- Highly efficient and flexible networking
- Routing, Overlay, Cloud-provider native
- IPv4, IPv6, NAT46
- Multi cluster routing

Service Load balancing:

- Highly scalable L3-L4 load balancing
- Kubernetes services (replaces kube-proxy)
- Multi-cluster
- Service affinity (prefer zones)

Container Security

- Identity-based network security
- API-aware security (HTTP, gRPC, Kafka, Cassandra, memcached, ..)
- DNS-aware policies
- Encryption
- SSL data visibility via kTLS

Visibility

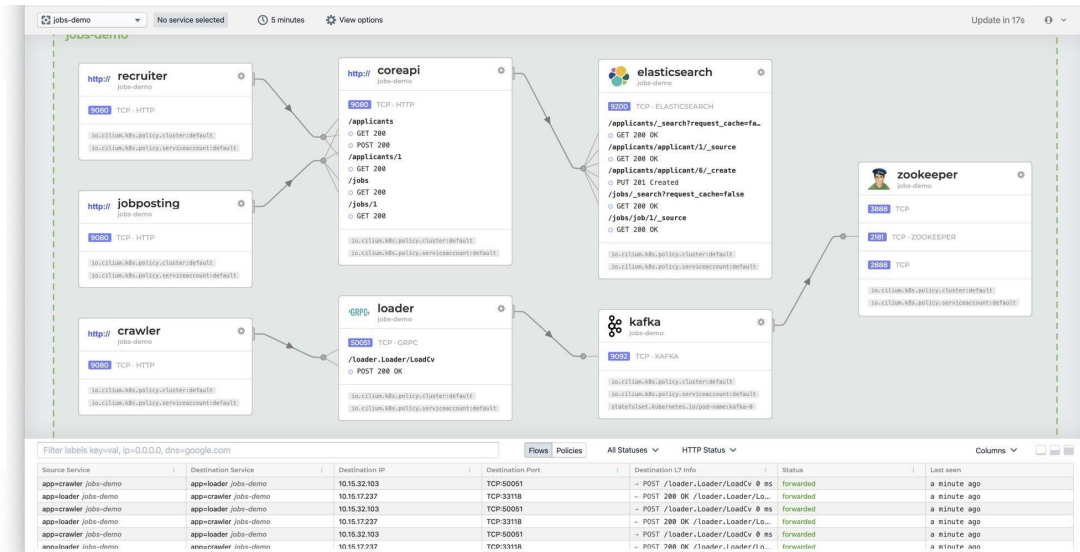
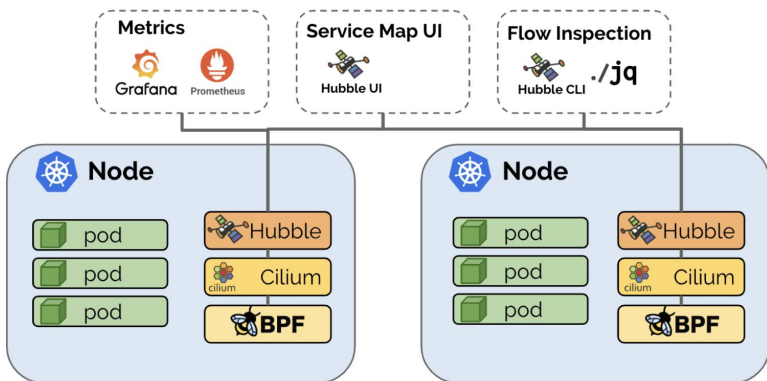
- Service topology map & live visualization
- Advanced network metrics & alerting

Servicemesh:

- Minimize overhead when injecting servicemesh sidecar proxies
- Istio integration

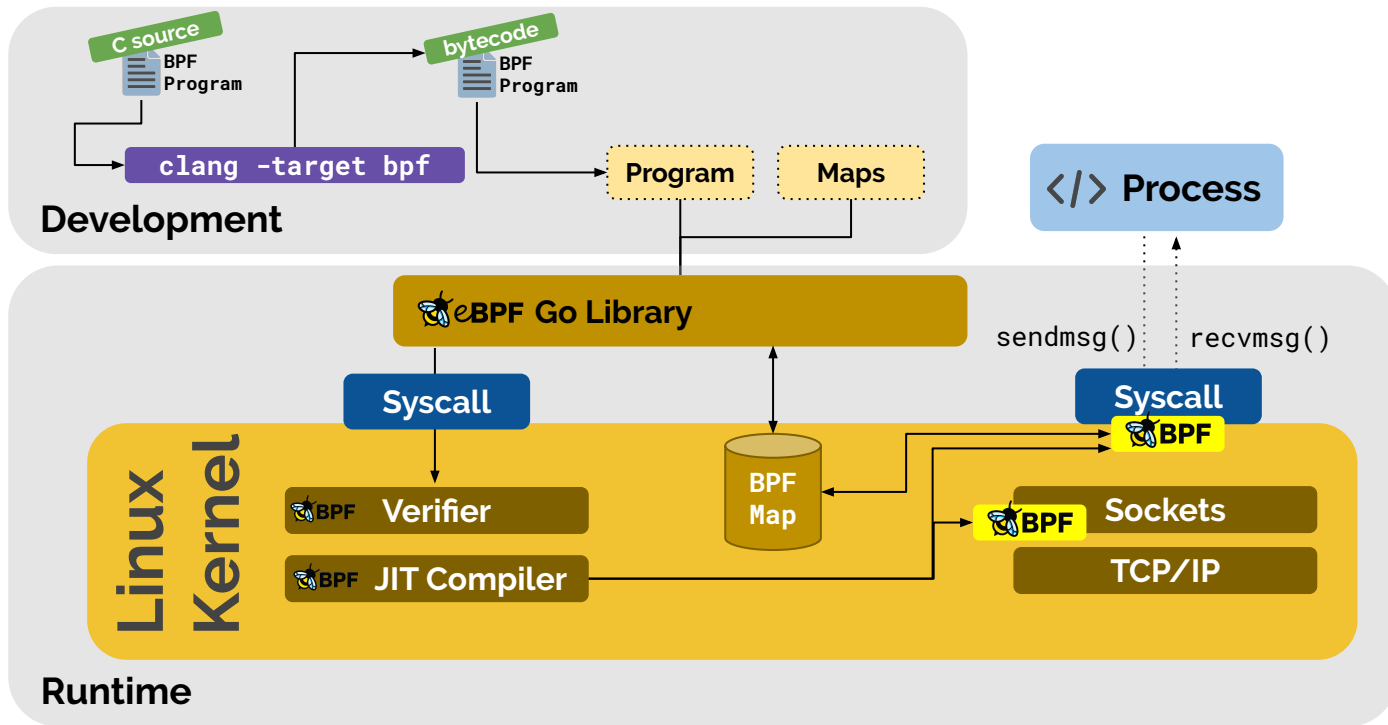
Hubble: eBPF Visibility for Kubernetes

cilium




```
# hubble observe --since=1m -t 17 -j \
| jq 'select(.l7.dns.rcode==3) | .destination.namespace + "/" + .destination.pod_name' \
| sort | uniq -c | sort -r
42 "starwars/jar-jar-binks-6f5847c97c-qmggv"
```



Go Development Toolchain



Outlook: Future of eBPF

 **eBPF** is turning the Linux kernel into a microkernel.

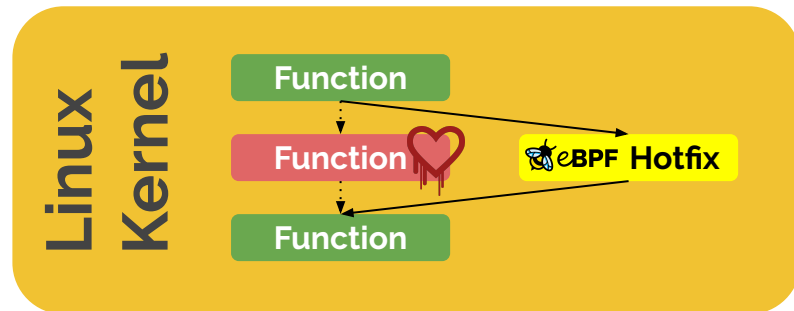
- An increasing amount of new kernel functionality is implemented with eBPF.
- 100% modular and composable.
- New additions can evolve at a rapid pace. Much quicker than normal kernel development.

Example: The linux kernel is not aware of containers and microservices (it only knows about namespaces).  Cilium is making the Linux kernel container and  Kubernetes aware.

 **eBPF** could enable the Linux kernel hotpatching we always dreamed about.

Problem:

- Linux kernel vulnerability requires to patch kernel.
- Rebooting 20'000 servers takes a very long time without risking extensive downtime.



Thank You

eBPF Maintainers

Daniel Borkmann, Alexei Starovoitov

Cilium Team

André Martins, Jarno Rajahalme, Joe Stringer,
John Fastabend, Maciej Kwiek, Martynas
Pumputis, Paul Chaignon, Quentin Monnet,
Ray Bejjani, Tobias Klauser

Facebook Team

Andrii Nakryiko, Andrey Ignatov, Jakub
Kicinski, Martin KaFai Lau, Roman Gushchin,
Song Liu, Yonghong Song

Google Team

Chenbo Feng, KP Singh, Lorenzo Colitti,
Maciej Żenczykowski, Stanislav Fomichev,

BCC & bpfttrace

Alastair Robertson, Brendan Gregg, Brenden
Blanco

Kernel Team

Björn Töpel, David S. Miller, Edward Cree,
Jesper Brouer, Toke Høiland-Jørgensen



cilium

- **BPF Getting Started Guide**
[BPF and XDP Reference Guide](#)
- **Cilium**
github.com/cilium/cilium
- **Twitter**
[@ciliumproject](#)
- **Contact the speaker**
[@tgraf__](#)