ISOVALENT

Buzzing Across the Cloud Native Landscape with eBPF

Bill Mulligan | @breakawaybilly
Community Pollinator, Isovalent


intel
inside
R

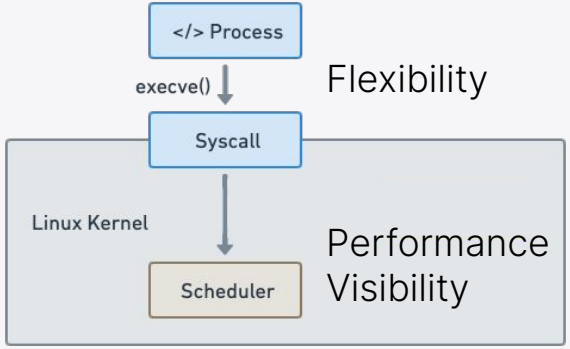
ISOVALENT

A Vision for eBPF Inside

@breakawaybilly

Kernel Space/User Space “Paradox”

- **Kernel**: System awareness, but lacks flexibility
- **User space**: Programmable, but no direct access to kernel structures, resources
- **Kernel modules**: Difficult, unsafe, not stable

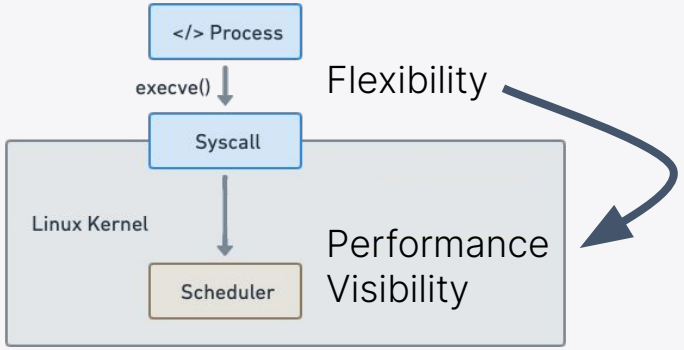


ISOVALENT

Kernel Space/User Space “Paradox”

Can we have **programmability** in the kernel?

How can this **benefit cloud native** environments?



ISOVALENT

What is  eBPF ?

@breakawaybilly

What is  eBPF ?

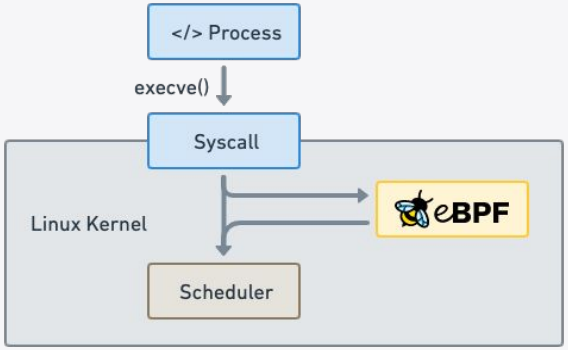
eBPF makes the Linux kernel programmable

ISOVALENT

eBPF

Makes the Linux kernel programmable in a secure and efficient way.

“What JavaScript is to the browser, eBPF is 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;
}
```

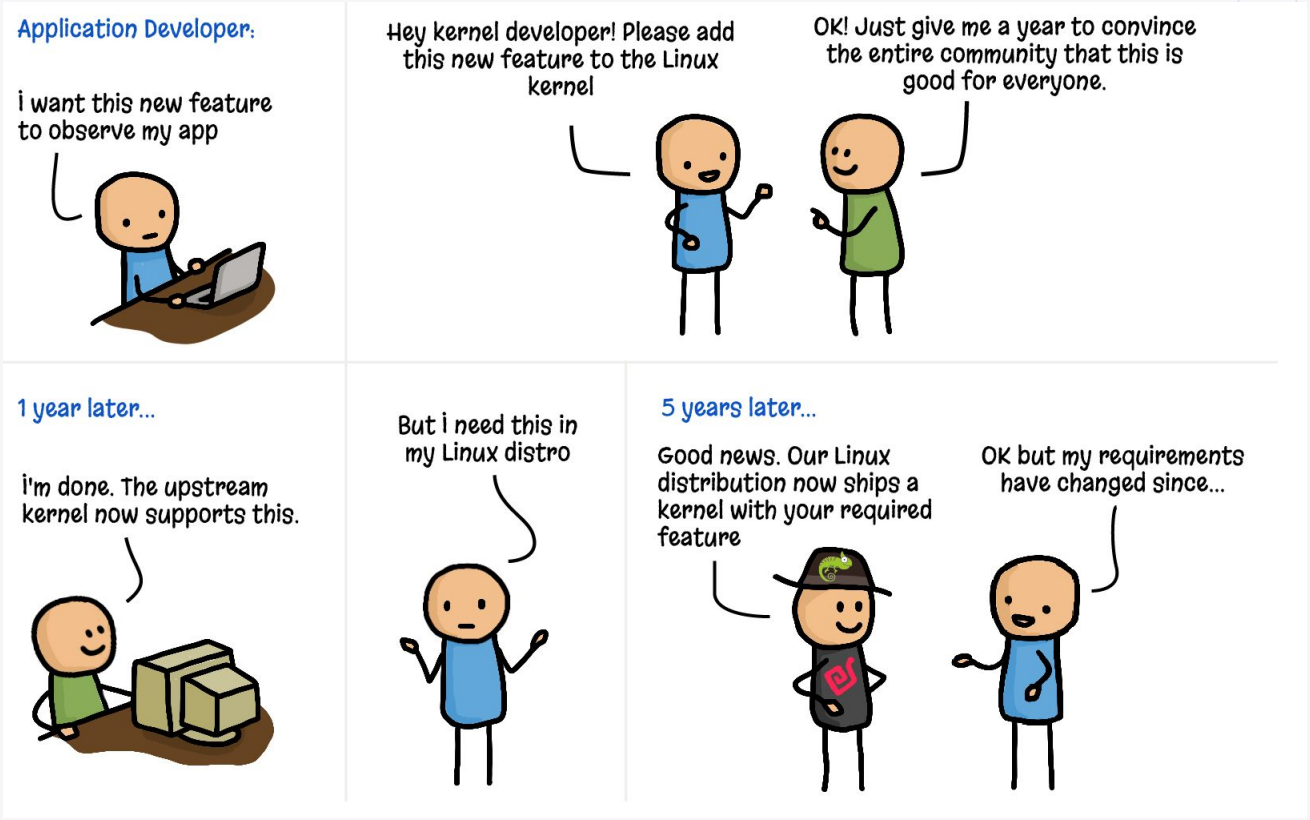
@breakawaybilly

"BPF has actually been really useful, and the real power of it is how it **allows people to do specialized code that isn't enabled until asked for**"

- Linus Torvalds

ISOVALENT

Dynamically Change Kernel Behavior



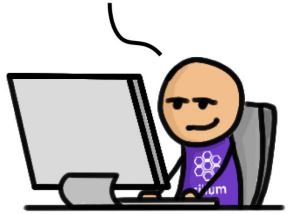
@breakawaybilly

Application Developer:

i want this new feature to observe my app

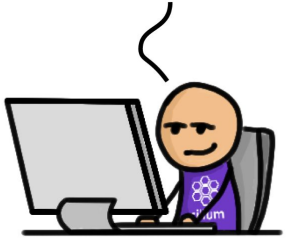
eBPF Developer:

OK! The kernel can't do this so let me quickly solve this with eBPF.



A couple of days later...

Here is a release of our eBPF project that has this feature now. BTW, you don't have to reboot your machine.

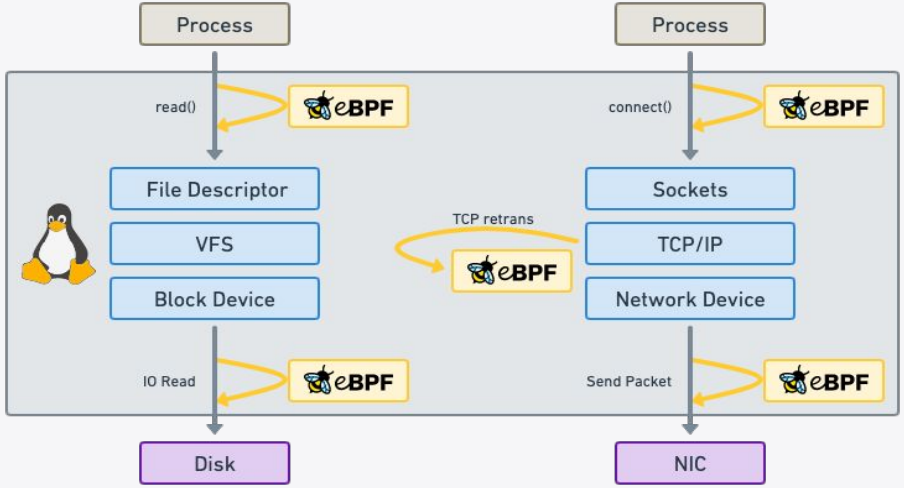


ISOVALENT

Run eBPF programs on events

Attachment points

- Kernel functions (kprobes)
- Userspace functions (uprobe)
- System calls
- Tracepoints
- Sockets (data level)
- Network devices (packet level)
- Network device (DMA level) [XDP]
- ...



@breakawaybilly

ISOVALENT

“**Superpowers for Linux**”

- Brendan Gregg

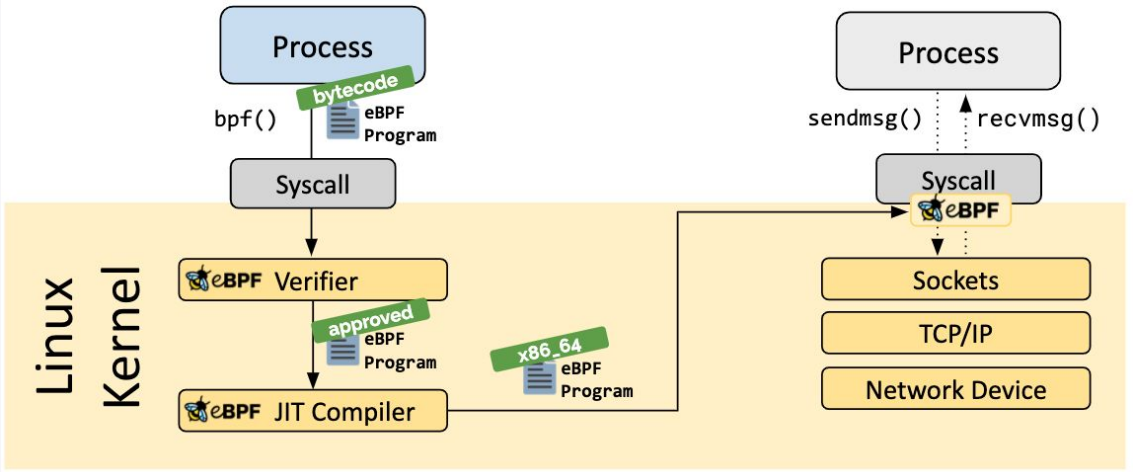
@breakawaybilly

ISOVALENT

Safe and Performant
Changes to Kernel Behavior

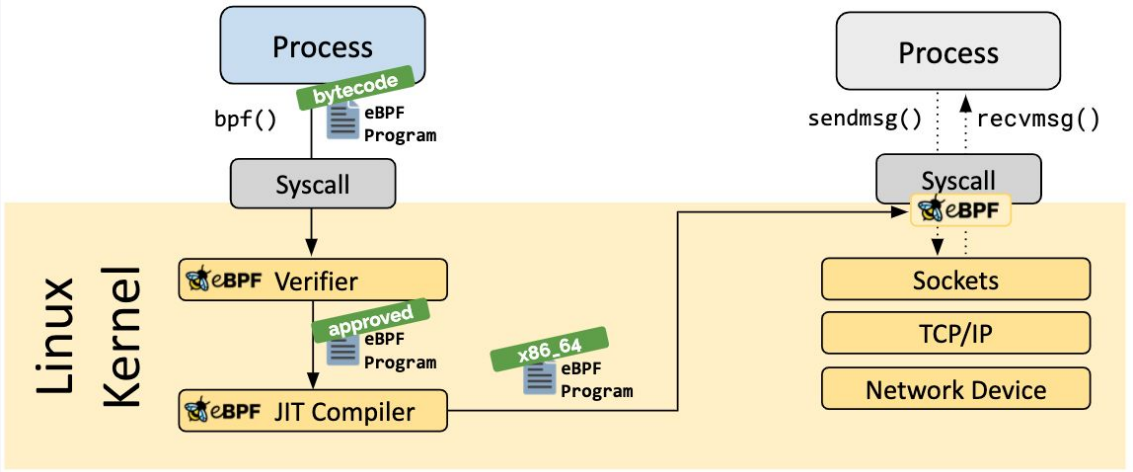
Safety: eBPF Verifier

- Programs will run to completion
- Does not crash or otherwise harm the system



Performance: In Kernel & JIT Compiler

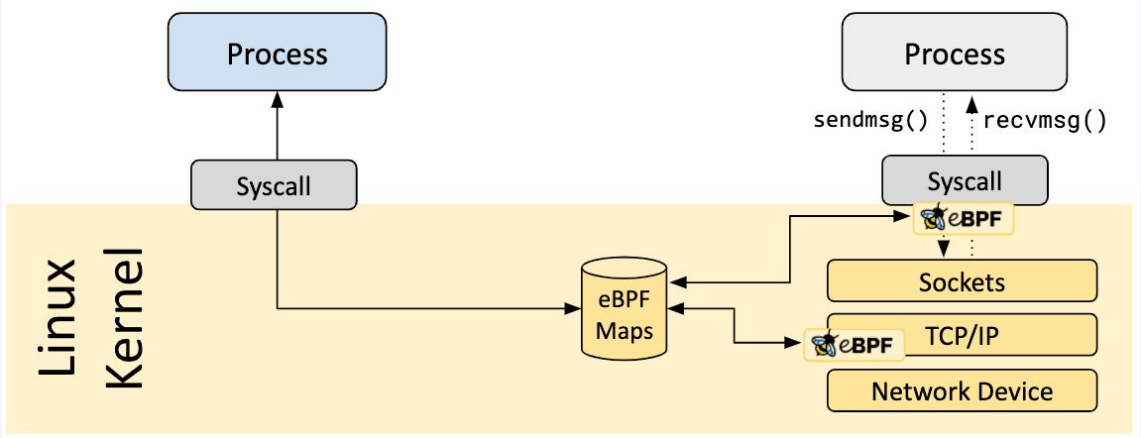
- Translate program bytecode into the machine specific instruction set to optimize execution speed of the program
- Runs as efficiently as natively compiled kernel code



Shared Memory: BPF maps

Between eBPF programs and between kernel and user space

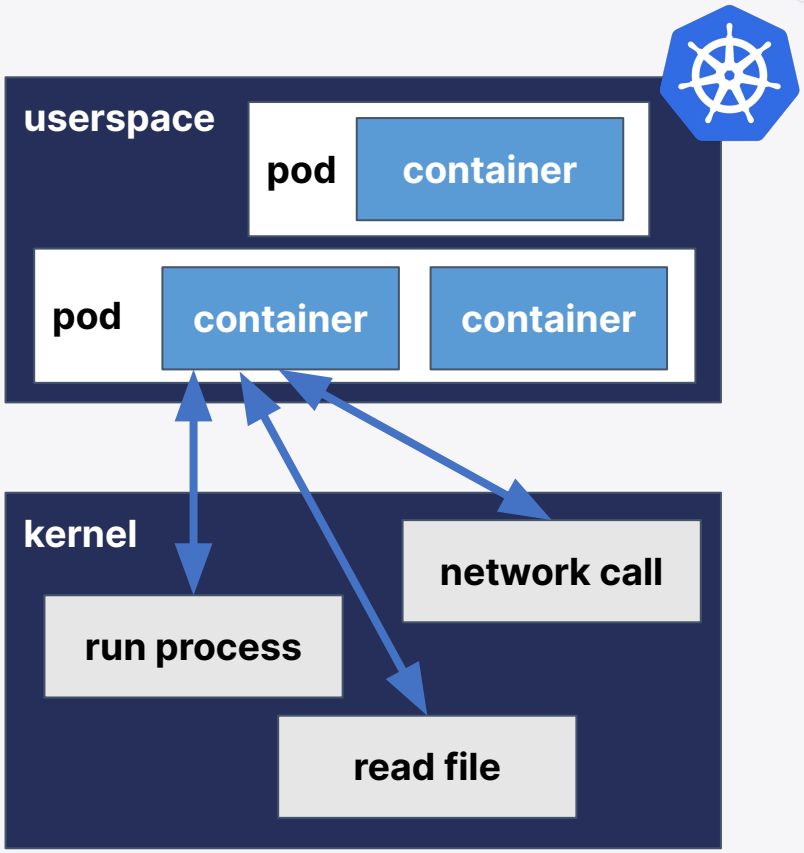
- No data loss
- More performant



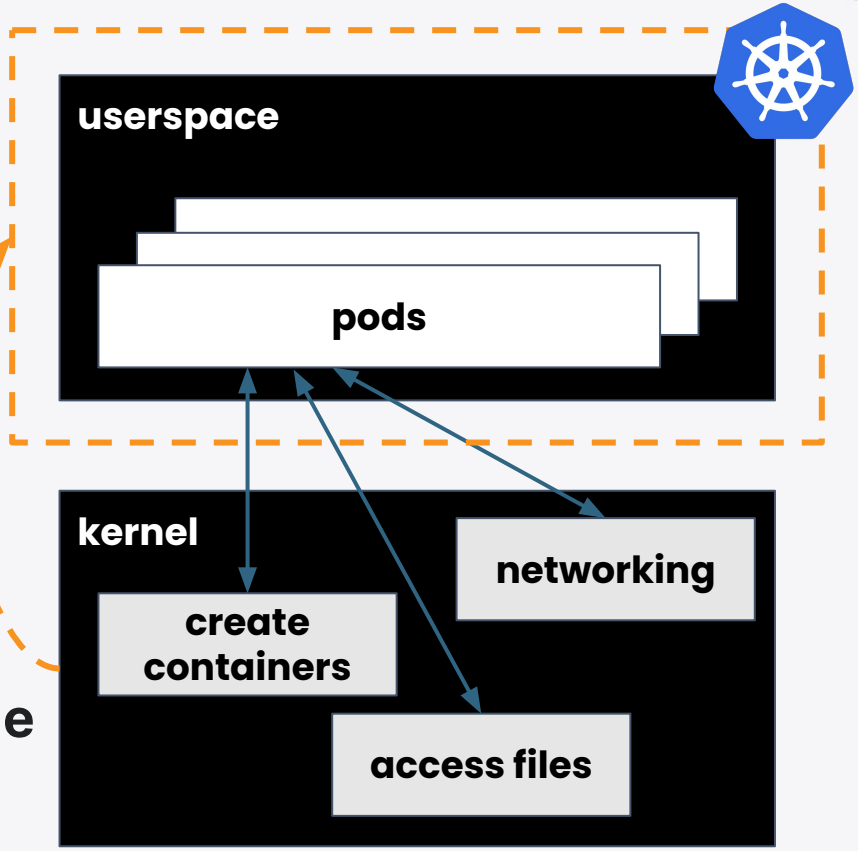
ISOVALENT

eBPF in Cloud Native

One kernel per host

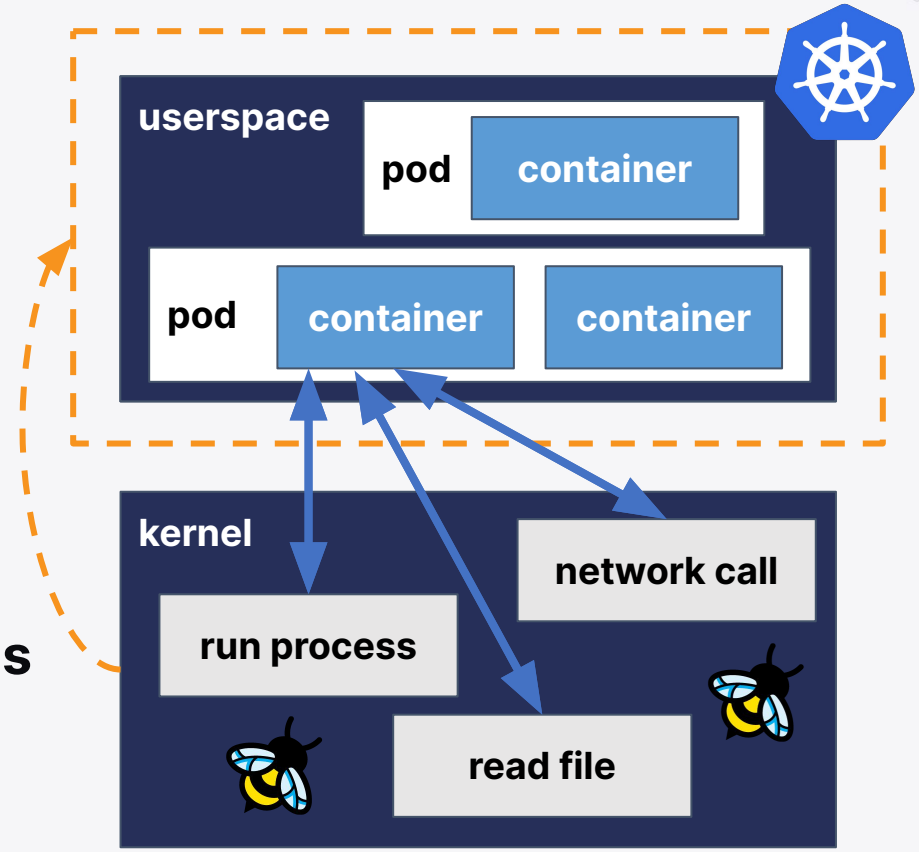


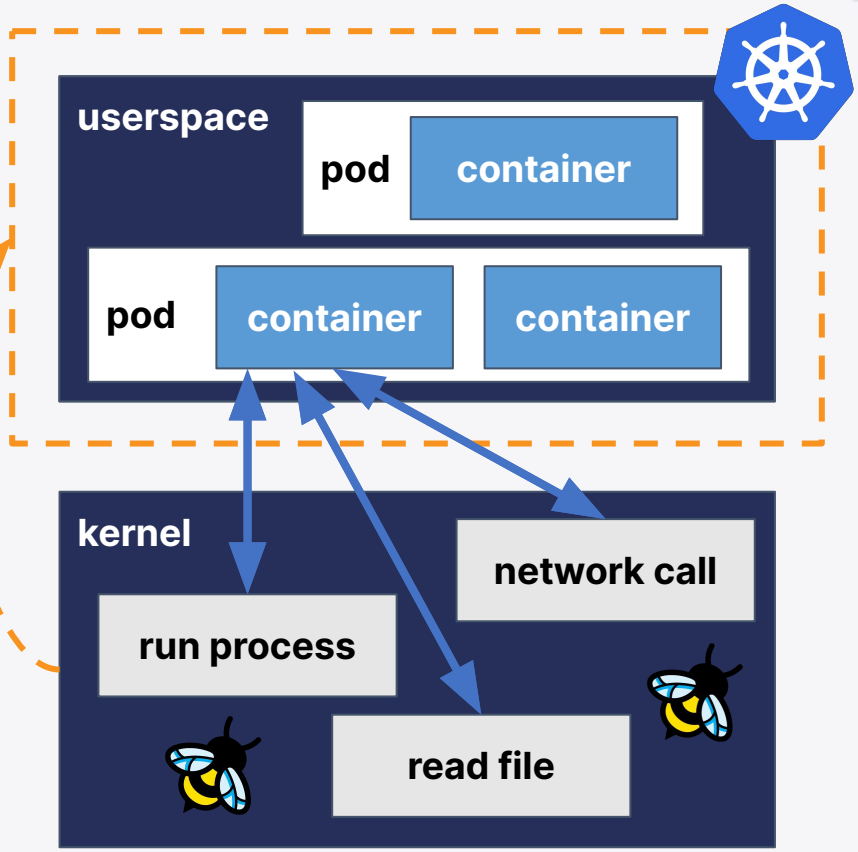
ISOVALENT



Kernel aware of everything on the host

eBPF programs can see everything





No changes to code or config

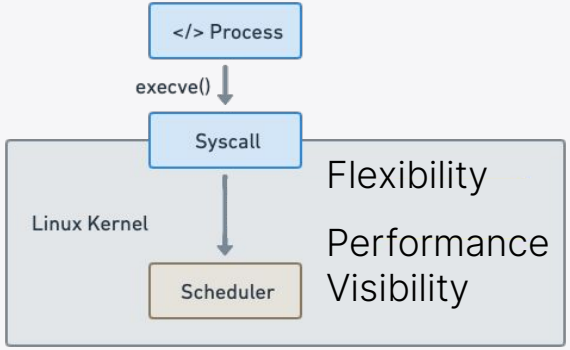
eBPF programs can see everything

eBPF in Kernel Space

In the kernel but **flexible**

Safety, performance, observability, and programmability

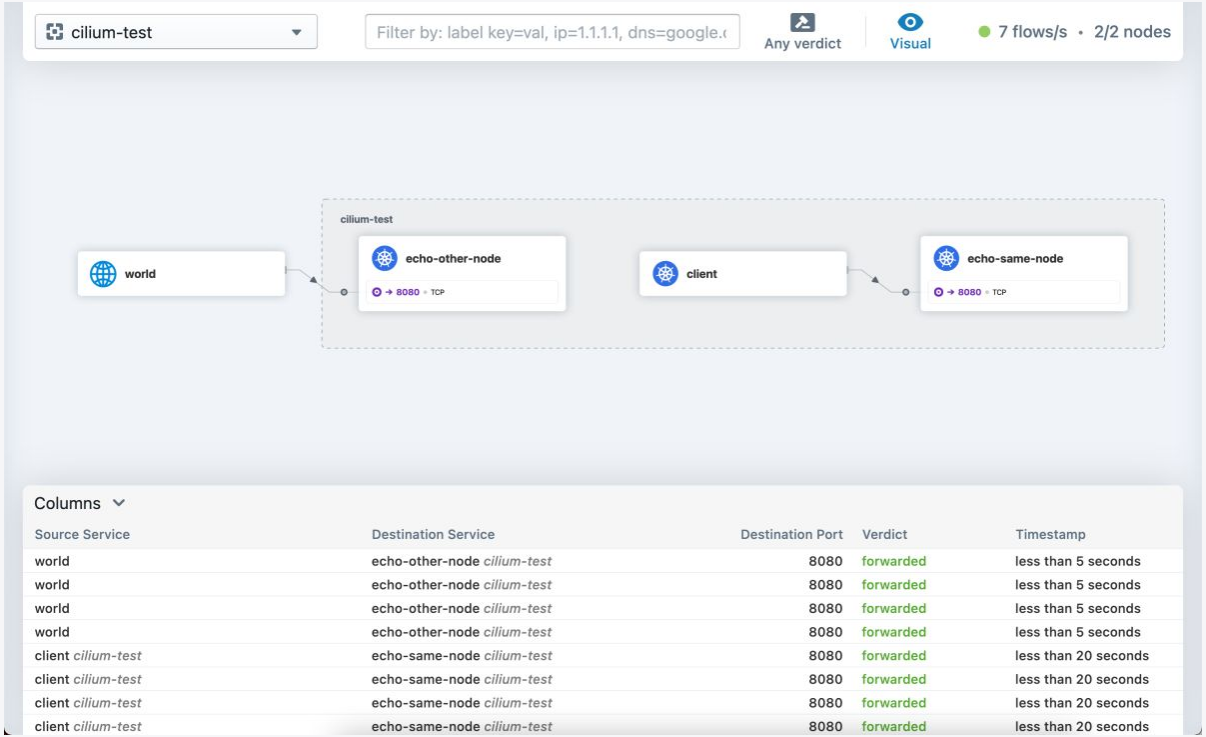
Available by default



ISOVALENT

Cloud Native
Use Cases

Networking

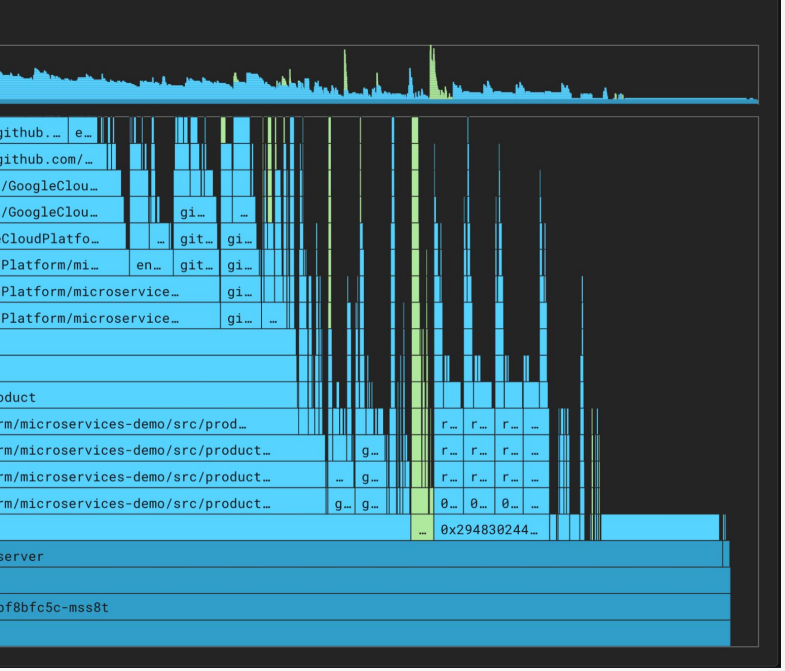


Columns

Source Service	Destination Service	Destination Port	Verdict	Timestamp
world	echo-other-node *cilium-test*	8080	forwarded	less than 5 seconds
world	echo-other-node *cilium-test*	8080	forwarded	less than 5 seconds
world	echo-other-node *cilium-test*	8080	forwarded	less than 5 seconds
world	echo-other-node *cilium-test*	8080	forwarded	less than 5 seconds
client *cilium-test*	echo-same-node *cilium-test*	8080	forwarded	less than 20 seconds
client *cilium-test*	echo-same-node *cilium-test*	8080	forwarded	less than 20 seconds
client *cilium-test*	echo-same-node *cilium-test*	8080	forwarded	less than 20 seconds
client *cilium-test*	echo-same-node *cilium-test*	8080	forwarded	less than 20 seconds

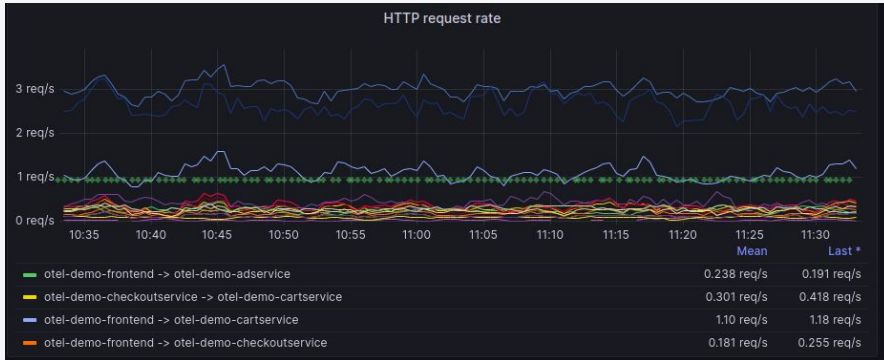
ISOVALENT

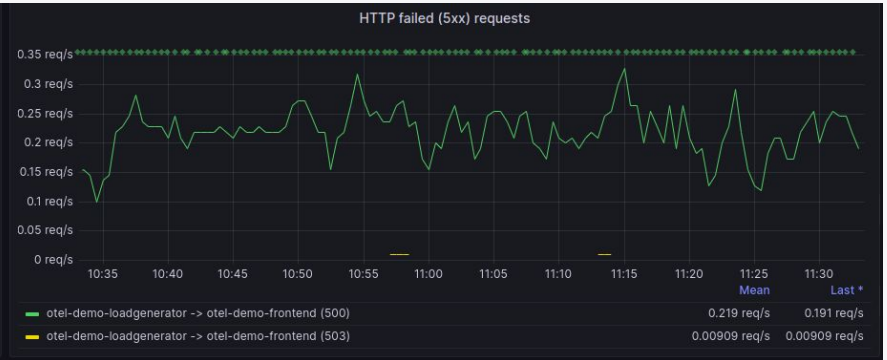
Observability

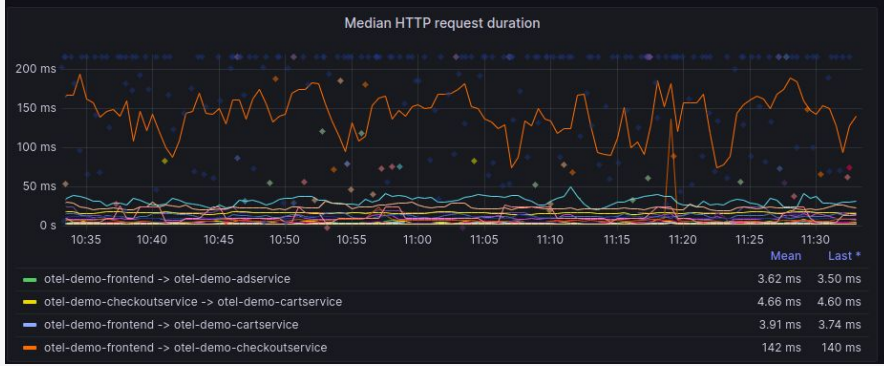


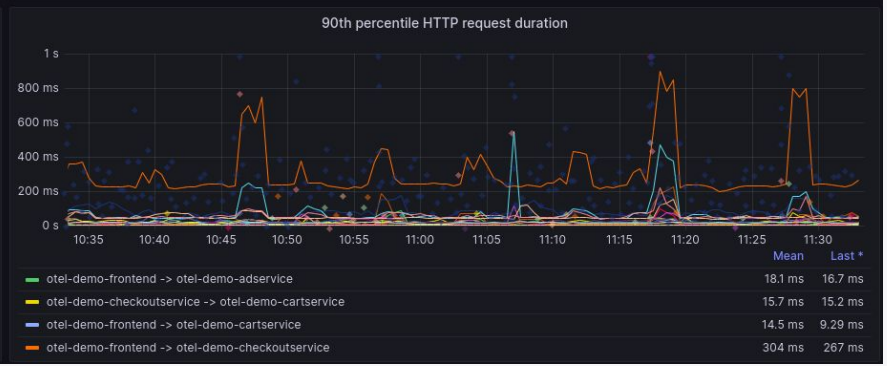
@breakawaybilly

Observability









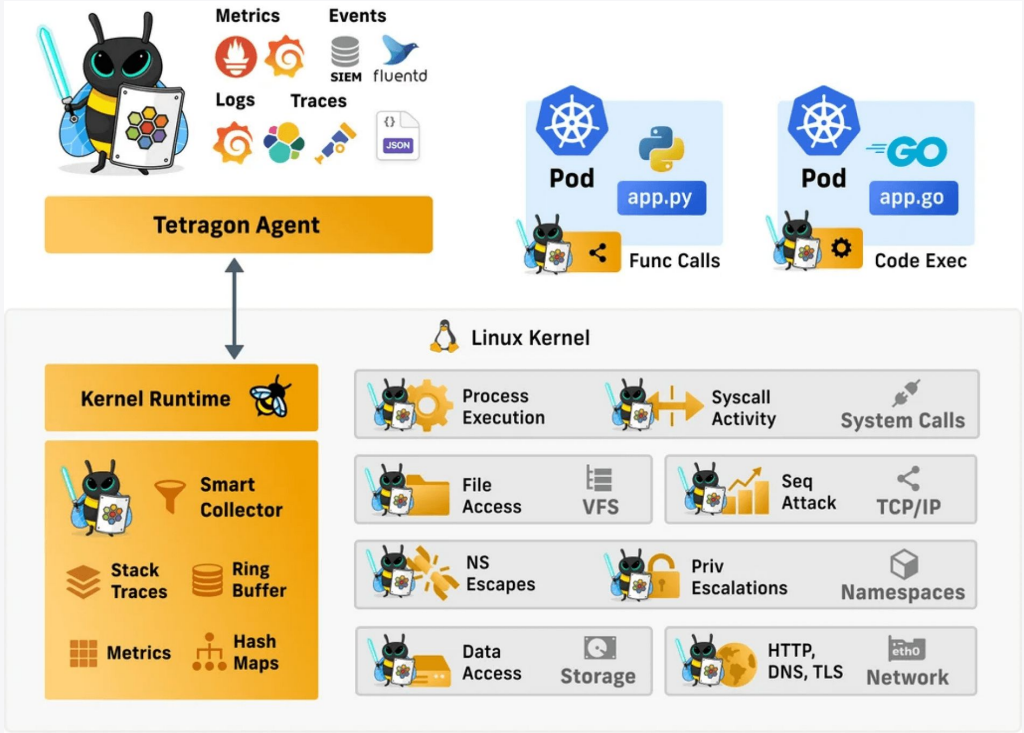
ISOVALENT

Security

Falco

KubeArmor

Tetragon



@breakawaybilly

ISOVALENT

Tracing and Profiling

Pyroscope

parca

@breakawaybilly

ISOVALENT

Buzzing Community

ISOVALENT

eBPF Landscape

Major Applications

bcc

Toolkit and library for efficient BPF-based kernel tracing

BCC is a toolkit for creating efficient kernel tracing and manipulation programs built upon eBPF, and includes several useful command-line tools and examples. BCC eases writing of eBPF programs for kernel instrumentation in C, includes a wrapper around LLVM, and front-ends in Python and Lua. It also provides a high-level library for direct integration into applications.

GITHUB

Cilium

eBPF-based Networking, Security, and Observability

Cilium is an open source project that provides eBPF-powered networking, security and observability. It has been specifically designed from the ground up to bring the advantages of eBPF to the world of Kubernetes and to address the new scalability, security and visibility requirements of container workloads.

GITHUB WEBSITE

bpftrace

High-level tracing language for Linux eBPF

bpftrace is a high-level tracing language for Linux eBPF. Its language is inspired by awk and C, and predecessor tracers such as DTrace and SystemTap. bpftrace uses LLVM as a backend to compile scripts to eBPF bytecode and makes use of BCC as a library for interacting with the Linux eBPF subsystem as well as existing Linux tracing capabilities and attachment points.

GITHUB WEBSITE

Falco

Cloud Native Runtime Security

Falco is a behavioral activity monitor designed to detect anomalous activity in applications. Falco audits a system at the Linux kernel layer with the help of eBPF. It enriches gathered data with other input streams such as container runtime metrics and Kubernetes metrics, and allows to continuously monitor and detect container, application, host, and network activity.

GITHUB WEBSITE

Katran

A high performance layer 4 load balancer

Katran is a C++ library and eBPF program to build a high-performance layer 4 load balancing forwarding plane. Katran leverages the XDP infrastructure from the Linux kernel to provide an in-kernel facility for fast packet processing. Its performance scales linearly with the number of NIC's receive queues and it uses RSS friendly encapsulation for forwarding to L7 load balancers.

GITHUB WEBSITE

Pixie

Scriptable observability for Kubernetes

Pixie is an open source observability tool for Kubernetes applications. Pixie uses eBPF to automatically capture telemetry data without the need for manual instrumentation. Developers can use Pixie to view the high-level state of their cluster (service maps, cluster resources, application traffic) and also drill down into more detailed views (pod state, flame graphs, individual full body application requests).

GITHUB WEBSITE

Calico

Pluggable eBPF-based networking and security for containers and Kubernetes

Calico Open Source is designed to simplify, scale, and secure container and Kubernetes networks. Calico's eBPF dataplane utilizes the power, speed, and efficiency of eBPF programs to deliver networking, load-balancing, and in-kernel security enforcement for your environment.

GITHUB WEBSITE

ISOVALENT

eBPF Landscape

Emerging

Pyroscope
Continuous Profiling Platform

Pyroscope is an open source project centered around continuous profiling, particularly in a Kubernetes context. It leverages eBPF as its core technology along with a custom storage engine to offer system-wide continuous profiling with minimal overhead as well as efficient storage and querying capabilities. We support Linux 4.9 and up thanks to CO-RE and libbpf.

GITHUB WEBSITE

eCapture
SSL/TLS capture tool using eBPF

eCapture is a Go language-written tool that can capture HTTPS/TLS plaintext without a CA certificate. It supports TLS encryption libraries such as openssl, boringssl, gnutls, and nspr. It can run on x86_64 CPU architectures with Linux kernel 4.18 or higher, and aarch64 CPU architectures with Linux/Android kernel 5.5 or higher, supporting both CO-RE and non-CO-RE modes without BTF.

GITHUB WEBSITE

Parca
Continuous Profiling Platform

Track memory, CPU, I/O bottlenecks broken down by method name, class name, and line number over time. Without complex overhead, in any language or framework. Using Parca's UI the data can be globally explored and analyzed using various visualizations to quickly and efficiently identify bottlenecks in code. Parca uses eBPF to collect profiling data and uses libbpf-go to interact with the kernel.

GITHUB WEBSITE

Hubble
Network, Service & Security Observability for Kubernetes using eBPF

Hubble is a fully distributed networking and security observability platform for cloud native workloads. It is built on top of Cilium and eBPF to enable deep visibility into the communication and behavior of services as well as the networking infrastructure in a completely transparent manner.

GITHUB WEBSITE

Tracee
Linux Runtime Security and Forensics using eBPF

Tracee uses eBPF technology to detect and filter operating system events, helping you expose security insights, detect suspicious behavior, and capture forensic indicators.

GITHUB

Tetragon
eBPF-based Security Observability & Runtime Enforcement

Tetragon provides eBPF-based transparent security observability combined with real-time runtime enforcement. The deep visibility is achieved without requiring application changes and is provided at low overhead thanks to smart Linux in-kernel filtering and aggregation logic built directly into the eBPF-based kernel-level collector. The embedded runtime enforcement layer is capable of performing access control on kernel functions, system calls and at other enforcement levels.

GITHUB

kubectl trace
Schedule bpftrace programs on your Kubernetes cluster

kubectl-trace is a kubectl plugin that allows for scheduling the execution of bpftrace(8) programs in Kubernetes clusters. kubectl-trace does not require installation of any components directly onto a Kubernetes cluster in order to execute bpftrace programs. When pointed to a cluster, it schedules a temporary job called trace-runner that executes bpftrace.

GITHUB

Inspektor Gadget
Introspecting and debugging Kubernetes applications using eBPF "gadgets"

Inspektor Gadget is a collection of tools (or gadgets) to debug and inspect Kubernetes resources and applications. It manages the packaging, deployment and execution of eBPF programs in a Kubernetes cluster, including many based on BCC tools, as well as some developed specifically for use in Inspektor Gadget. It automatically maps low-level kernel primitives to high-level Kubernetes resources, making it easier and quicker to find the relevant information.

GITHUB WEBSITE

Sysinternals Sysmon for Linux
Security Observability

Sysmon for Linux is a tool that monitors and logs system activity including process lifetime, network connections, file system writes, and more. Sysmon works across reboots and supports advanced filtering to help identify malicious activity as well as how intruders and malware operate on your network.

GITHUB

pwru
eBPF-based Linux kernel network packet tracer

pwru is an eBPF-based tool for tracing network packets in the Linux kernel with advanced filtering capabilities. It allows fine-grained introspection of kernel state to facilitate debugging network connectivity issues.

GITHUB

Caretta
eBPF based Kubernetes service map

Caretta is a Kubernetes service map that uses eBPF to trace network traffic between pods. It can be used to visualize the network traffic between services in a Kubernetes cluster, and gain additional insights into the network traffic and the relationships between services.

GITHUB

BumbleBee
OCI compliant eBPF tooling

BumbleBee simplifies building eBPF tools and allows you to package, distribute, and run eBPF programs using OCI images. It allows you to just focus on the eBPF portion of your code and BumbleBee automates away the boilerplate, including the userspace code.

GITHUB WEBSITE

ply
A dynamic tracer for Linux

ply is a dynamic tracer for Linux which is built upon eBPF. It has been designed with embedded systems in mind, is written in C and all that ply needs to run is libc and a modern Linux kernel with eBPF support, meaning, it does not depend on LLVM for its program generation. It has a C-like syntax for writing scripts and is heavily inspired by awk(1) and dtrace(1).

GITHUB WEBSITE

KubeArmor
Container-aware Runtime Security Enforcement System

KubeArmor is a container-aware runtime security enforcement system that restricts the behavior (such as process execution, file access, networking operation, and resource utilization) of containers at the system level, using LSMs and eBPF.

GITHUB WEBSITE

Merbridge
Use eBPF to speed up your Service Mesh like crossing an Einstein-Rosen Bridge

Merbridge is designed to make traffic interception and forwarding more efficient for service mesh. With Merbridge, developers can use eBPF instead of iptables to accelerate their service mesh without any additional operations or code changes. Currently, Merbridge already supports Istio, Linkerd, and Kuma.

GITHUB WEBSITE

DeepFlow
Highly Automated Observability Platform powered by eBPF

DeepFlow is a highly automated observability platform built for cloud native developers. Based on eBPF, DeepFlow innovatively implements an automated distributed tracing mechanism: AutoTracing. Microservice processes, service mesh sidecars, and network interfaces along the way are included as tracing spans, for every distributed transaction, without any code instrumentation. DeepFlow can automatically generate golden RED metrics for any process in cloud native environment.

GITHUB WEBSITE

wachy
UI for interactive eBPF-based userspace performance debugging

Wachy is a profiler that uses eBPF to trace arbitrary compiled binaries and functions at runtime. It aims to make eBPF uprobe-based debugging much easier to use by displaying traces in a UI next to the source code, and allowing interactive drilldown analysis.

GITHUB WEBSITE

Kindling
eBPF-based Cloud Native Monitoring & Profiling Tool

Kindling is a monitoring tool that aims to help users understand the execution behavior of programs from kernel space to user space to pinpoint the root cause of critical incidents. It can obtain L4/L7 network performance metrics and build service maps. Kindling implements a mechanism, Trace Profiling, that can display how each trace is executing on-CPU with thread-level flame graph, and how it is slowed down by off-CPU events with related metrics.

GITHUB WEBSITE

Pulsar
A modular runtime security framework for the IoT

Pulsar is an event-driven framework for monitoring the activity of Linux devices. It allows you to collect runtime activity events from the Linux kernel through its modules and evaluate each event against your own set of security policies. Powered by eBPF and written in Rust, Pulsar is lightweight and safe by design.

GITHUB WEBSITE

Kepler
Kubernetes-based Efficient Power Level Exporter

Kepler (Kubernetes-based Efficient Power Level Exporter) is a Prometheus exporter. It uses eBPF to probe CPU performance counters and Linux kernel tracepoints. These data and stats from cgroup and sysfs are fed into ML models to estimate energy consumption by Pods.

GITHUB WEBSITE

eunomia-bpf
eBPF programs in a WASM module or JSON

Eunomia-bpf is a dynamic loading library, based on libbpf, and a compiler toolchain. Eunomia-bpf simplifies building eBPF tools and allows you to package, distribute, and run eBPF programs in JSON format or as a WASM module. With eunomia-bpf, you can write kernel eBPF code and automatically expose your data from the kernel and interact with eBPF program in user space with a WASM runtime.

GITHUB WEBSITE

L3AF
Complete lifecycle management of eBPF programs

L3AF is a platform to launch and manage eBPF programs in distributed environments. L3AF empowers users to compose multiple eBPF programs together to solve unique problems in different environments. Using the APIs provided by L3AF, these eBPF programs can be reconfigured, updated, inspected, and reordered on-the-fly. L3AF also provides configurable metrics for the eBPF programs it has launched.

GITHUB WEBSITE

Apache SkyWalking
APM, Application Performance Monitoring System

Apache SkyWalking is an application performance monitor tool for distributed systems, especially designed for microservices, cloud native and container-based (Kubernetes) architectures. SkyWalking Rover is an agent in the SkyWalking ecosystem, as a metrics collector and profiler powered by eBPF to diagnose CPU, I/O and L4/L7(TLS) network performance. Also, Rover provides add-on events for spans in the distributed tracing.

GITHUB WEBSITE

LoxiLB
eBPF based cloud-native load-balancer for 5G Edge

LoxiLB is an open-source cloud-native "external" service load-balancer for cloud-native 5G/edge workloads written from scratch using eBPF as its core-engine and based on Go Language. LoxiLB turns Kubernetes network load balancing for 5G/Edge services into high speed, flexible and programmable LB services.

GITHUB WEBSITE

SSHLog
eBPF SSH session monitoring

SSHLog is a Linux daemon written in C++ and Python that monitors OpenSSH servers via eBPF. The agent passively records all SSH session activity (commands and output) to log files for any connecting user. Administrators can also share an SSH session with any logged in user. Actions may be triggered based on SSH behavior such as posting a Slack message when a remote user attempts to gain root access.

GITHUB WEBSITE

bpfd
A system daemon and Kubernetes operator for managing eBPF programs

bpfd is a system daemon aimed at simplifying the deployment and management of eBPF programs. It's goal is to enhance the developer-experience as well as provide features to improve security, visibility and program-cooperation. bpfd includes a Kubernetes operator to bring those same features to Kubernetes, allowing users to safely deploy eBPF via custom resources across nodes in a cluster.

GITHUB WEBSITE

blixt
Layer 4 Kubernetes load-balancer

Blixt is a layer 4 load-balancer for Kubernetes. It has a control-plane implemented using Gateway API and a data-plane built using eBPF and Rust.

GITHUB

ISOVALENT

eBPF Landscape

Major Infrastructure

Linux Kernel

eBPF Runtime

The Linux kernel contains the eBPF runtime required to run eBPF programs. It implements the bpf(2) system call for interacting with programs, maps, BTF and various attachment points where eBPF programs can be executed from. The kernel contains a eBPF verifier in order to check programs for safety and a JIT compiler to translate programs to native machine code. User space tooling such as bpftool and libbpf are also maintained as part of the upstream kernel.

GIT TREES · WEBSITE · PATCHES · MAILING LIST · OFFICE HOURS · CI · DOCS

LLVM Compiler

eBPF Backend

The LLVM compiler infrastructure contains the eBPF backend required to translate programs written in a C-like syntax to eBPF instructions. LLVM generates eBPF ELF files which contain program code, map descriptions, relocation information and BTF meta data. These ELF files contain all necessary information for eBPF loaders such as libbpf to prepare and load programs into the Linux kernel. The LLVM project also contains additional developer tools such as an eBPF object file disassembler.

GITHUB · WEBSITE · COMPILER EXPLORER · BUGTRACKER · PATCHES

GCC Compiler

eBPF Backend

The GCC compiler comes with an eBPF backend starting from GCC 10. Up to that point, LLVM has been the only compiler which supports generating eBPF ELF files. The GCC port is roughly equivalent to the LLVM eBPF support. There are some missing bits of functionality but the GCC community is working to close these gaps over time. GCC also contains eBPF binutils as well as eBPF gdb support for debugging of eBPF code that is traditionally consumed by the Linux kernel. Included as part of this is an eBPF simulator for gdb.

GIT REPO · WEBSITE · COMPILER EXPLORER · BUGTRACKER · MAILING LIST · DOCS

Emerging

eBPF for Windows

eBPF Runtime

The eBPF for Windows project is a work-in-progress that allows using existing eBPF toolchains and APIs familiar in the eBPF ecosystem to be used on top of Windows. That is, this project takes existing eBPF projects as submodules and adds the layer in between to make them run on top of Windows.

GITHUB · WEBSITE · OFFICE HOURS · SLACK CHANNEL

uBPF

Userspace eBPF Runtime

An eBPF runtime that permits execution of eBPF programs in user mode, with support for an interpreter as well as JIT compilation of eBPF programs on x86-64 and ARM64 architectures. This project supports running on Windows, macOS, and Linux platforms.

GITHUB

rbpf

User Space eBPF Runtime

Running in user space, rbpf provides a cross-platform eBPF interpreter and a JIT-compiler for x86-64, implemented in Rust. It was initially a port of uBPF to Rust.

GITHUB · CRATE · DOCS

hBPF

eBPF in Hardware

An extended Berkley Packet Filter CPU implemented in hardware on FPGA. In contrast to classic HDL languages like Verilog or VHDL, Migen/LiteX (both based on Python) where used. Supports custom extensions to 'call' opcode and includes full test suite for each opcode for included emulator and simulator as well as for included hardware targets.

GITHUB · WEBSITE

PREVAIL

eBPF Verifier

A polynomial-time eBPF verifier supporting bounded loops based on abstract interpretation.

GITHUB

BPF Conformance

eBPF Conformance Testing Framework

A conformance testing framework for eBPF runtime implementations. It provides a set of tests that can be used to verify that an eBPF implementation is compliant with the eBPF specification.

GITHUB

eBPF Libraries

Golang

eBPF is designed as a pure Go library that provides utilities for loading, compiling, and debugging eBPF programs. It has minimal external dependencies and is intended to be used in long running processes.

libbpfgo is a Go wrapper around libbpf. It supports BPF CO-RE and its goal is to be a complete implementation of libbpf APIs. It uses CGo to call into linked versions of libbpf.

C++

libbpf is a C/C++ based library which is maintained as part of the upstream Linux kernel. It contains an eBPF loader which takes over processing LLVM generated eBPF ELF files for loading into the kernel. libbpf received a major boost in capabilities and sophistication and closed many existing gaps with BCC as a library. It also supports important features not available in BCC such as global variables and BPF skeletons.

Rust

libbpf-rs is a safe, idiomatic, and opinionated wrapper API around libbpf written in Rust. libbpf-rs, together with libbpf-cargo (libbpf cargo plugin) allows to write 'compile once run everywhere' (CO-RE) eBPF programs.

redbpf is a Rust eBPF toolchain that contains a collection of Rust libraries to work with eBPF programs.

aya is an eBPF library built with a focus on operability and developer experience. It allows for both eBPF programs and their userspace programs to be written in Rust.

eBPF Auxiliary Libraries

libxdp

Utilities for use with XDP

libxdp is an XDP-specific library that sits on top of libbpf and implements a couple of XDP features. It supports loading of multiple programs to run in sequence on the same interface, and it contains helper functions for configuring AF_XDP sockets as well as reading and writing packets from these sockets.

GITHUB

@breakawaybilly

ISOVALENT

Kernel Community

```
From: Daniel Borkmann <dborkman@redhat.com>
To: davem@davemloft.net
Cc: ast@plumgrid.com, netdev@vger.kernel.org
Subject: [PATCH net-next v4 0/9] BPF updates
Date: Fri, 28 Mar 2014 18:58:17 +0100   [thread overview]
Message-ID: <1396029506-16776-1-git-send-email-dborkman@redhat.com> (raw)

We sat down and have heavily reworked the whole previous patchset
from v10 [1] to address all comments/concerns. This patchset therefore
*replaces* the internal BPF interpreter with the new layout as
discussed in [1], and migrates some exotic callers to properly use the
BPF API for a transparent upgrade. All other callers that already use
the BPF API in a way it should be used, need no further changes to run
the new internals. We also removed the sysctl knob entirely, and do not
expose any structure to userland, so that implementation details only
reside in kernel space. Since we are replacing the interpreter we had
to migrate seccomp in one patch along with the interpreter to not break
anything. When attaching a new filter, the flow can be described as
following: i) test if jit compiler is enabled and can compile the user
BPF, ii) if so, then go for it, iii) if not, then transparently migrate
the filter into the new representation, and run it in the interpreter.
Also, we have scratched the jit flag from the len attribute and made it
as initial patch in this series as Pablo has suggested in the last
feedback, thanks. For details, please refer to the patches themselves.

We did extensive testing of BPF and seccomp on the new interpreter
itself and also on the user ABIs and could not find any issues; new
performance numbers as posted in patch 8 are also still the same.

Please find more details in the patches themselves.

For all the previous history from v1 to v10, see [1]. We have decided
to drop the v11 as we have pedantically reworked the set, but of course,
included all previous feedback.
```

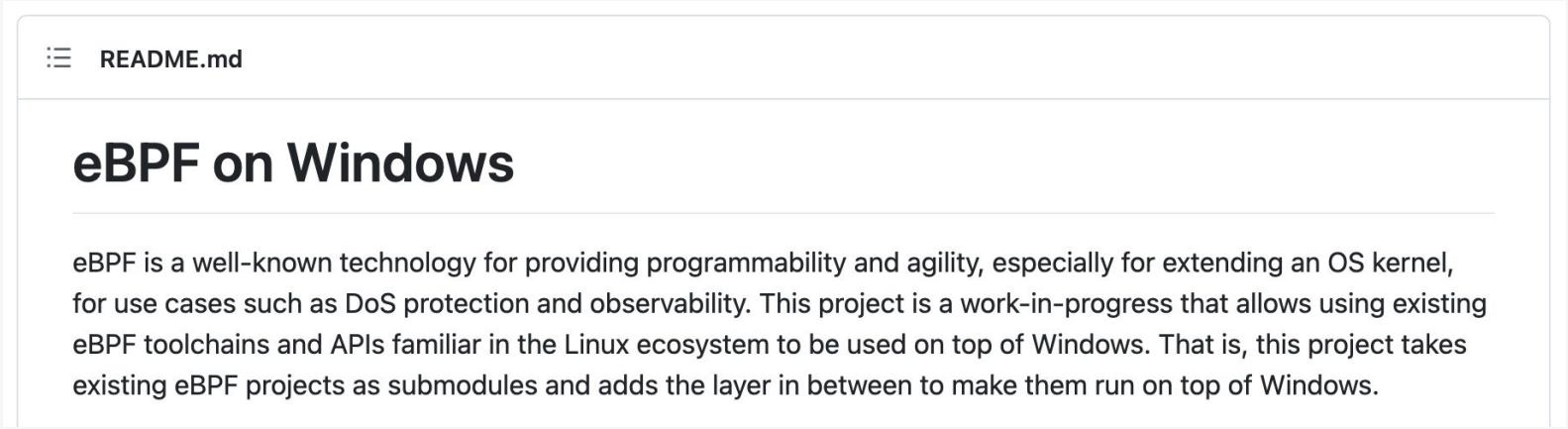
@breakawaybilly

eBPF Acquisitions

- Flowmill ➡ Splunk (November 2020)
- Kinvolk ➡ Microsoft (April 2021)
- Pixie ➡ New Relic (July 2021)
- Cmd ➡ Elastic (October 2021)
- Optimyze ➡ Elastic (August 2021)
- Seekret ➡ Datadog (August 2022)
- Pyroscope ➡ Grafana (March 2023)

ISOVALENT

eBPF for Windows

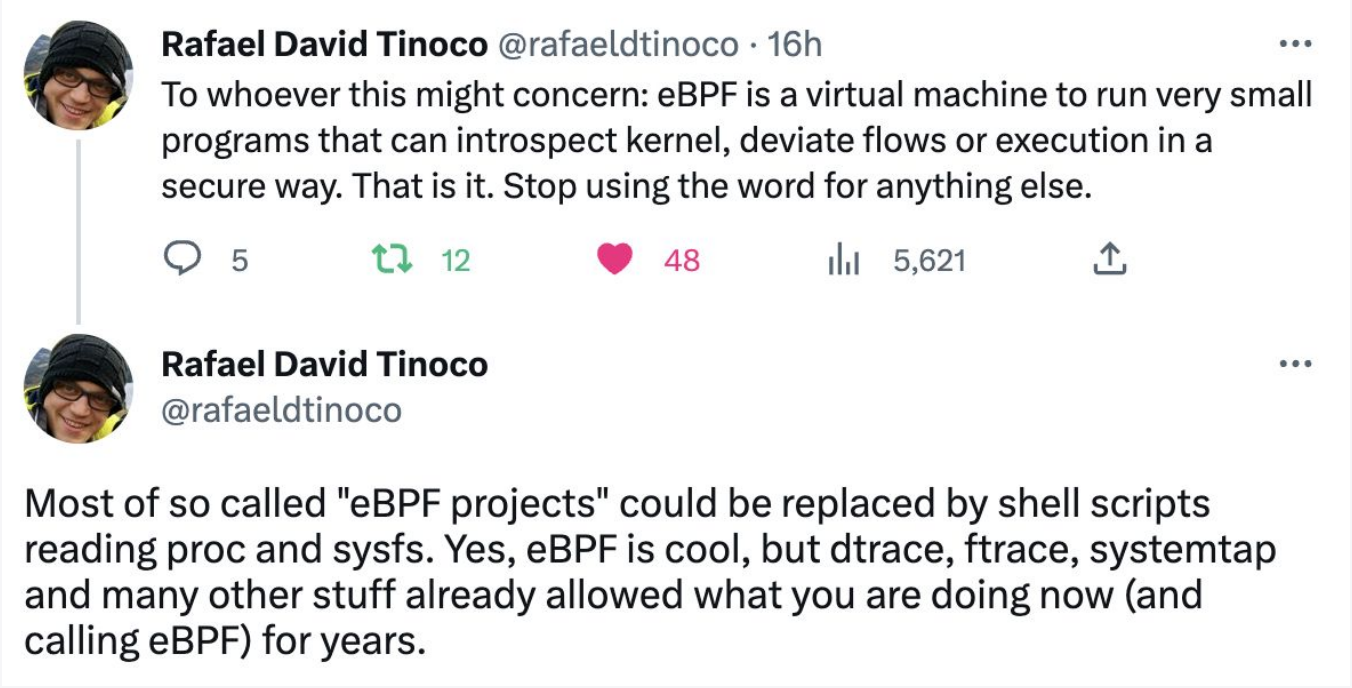


README.md

eBPF on Windows

eBPF is a well-known technology for providing programmability and agility, especially for extending an OS kernel, for use cases such as DoS protection and observability. This project is a work-in-progress that allows using existing eBPF toolchains and APIs familiar in the Linux ecosystem to be used on top of Windows. That is, this project takes existing eBPF projects as submodules and adds the layer in between to make them run on top of Windows.

Still not a magic bullet



Rafael David Tinoco @rafaeldtinoco · 16h

To whoever this might concern: eBPF is a virtual machine to run very small programs that can introspect kernel, deviate flows or execution in a secure way. That is it. Stop using the word for anything else.

5 12 48 5,621

Rafael David Tinoco
@rafaeldtinoco

Most of so called "eBPF projects" could be replaced by shell scripts reading proc and sysfs. Yes, eBPF is cool, but dtrace, ftrace, systemtap and many other stuff already allowed what you are doing now (and calling eBPF) for years.

@breakawaybilly

ISOVALENT

Where to go
Next

ISOVALENT

eBPF Wikipedia

eBPF

Add languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

eBPF, the **extended Berkeley Packet Filter** (often referred to by the pseudo-acronym **BPF**)[2][5] is a technology that can run sandboxed programs in a privileged context such as the operating system kernel.[6] It is used to safely and efficiently extend the capabilities of the kernel at runtime without requiring to change kernel source code or load kernel modules.[7] Safety is provided through an in-kernel verifier which performs static code analysis and rejects programs which crash, hang or otherwise interfere with the kernel negatively.[8][9] Examples of programs that are automatically rejected are programs without strong exit guarantees (i.e. for/while loops without exit conditions) and programs dereferencing pointers without safety-checks.[10] Loaded programs which passed the verifier are either interpreted or in-kernel JIT compiled for native execution performance. The execution model is event-driven and with few exceptions run-to-completion,[2] meaning, programs can be attached to various hook points in the operating system kernel and are run upon triggering of an event. eBPF use cases include (but are not limited to) networking such as XDP, tracing and security subsystems.[6] Given eBPF's efficiency and flexibility opened up new possibilities to solve production issues, Brendan Gregg famously coined eBPF as "superpowers for Linux".[11] Linus Torvalds expressed that "BPF has actually been really useful, and the real power of it is how it allows people to do specialized code that isn't enabled until asked for".[12] Due to its success in Linux, the eBPF runtime has been ported to other operating systems such as Windows.[4]

eBPF

Original author(s)	Alexei Starovoitov, Daniel Borkmann[1][2]
Developer(s)	Open source community, Meta, Google, Isovalent, Microsoft, Netflix[1]
Initial release	2014; 9 years ago[3]
Repository	Linux: git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/ Windows: github.com/Microsoft/ebpf-for-windows/
Written in	C
Operating system	Linux, Windows[4]
Type	Runtime system, Sandboxing
License	Linux: GPL Windows: MIT License
Website	ebpf.io

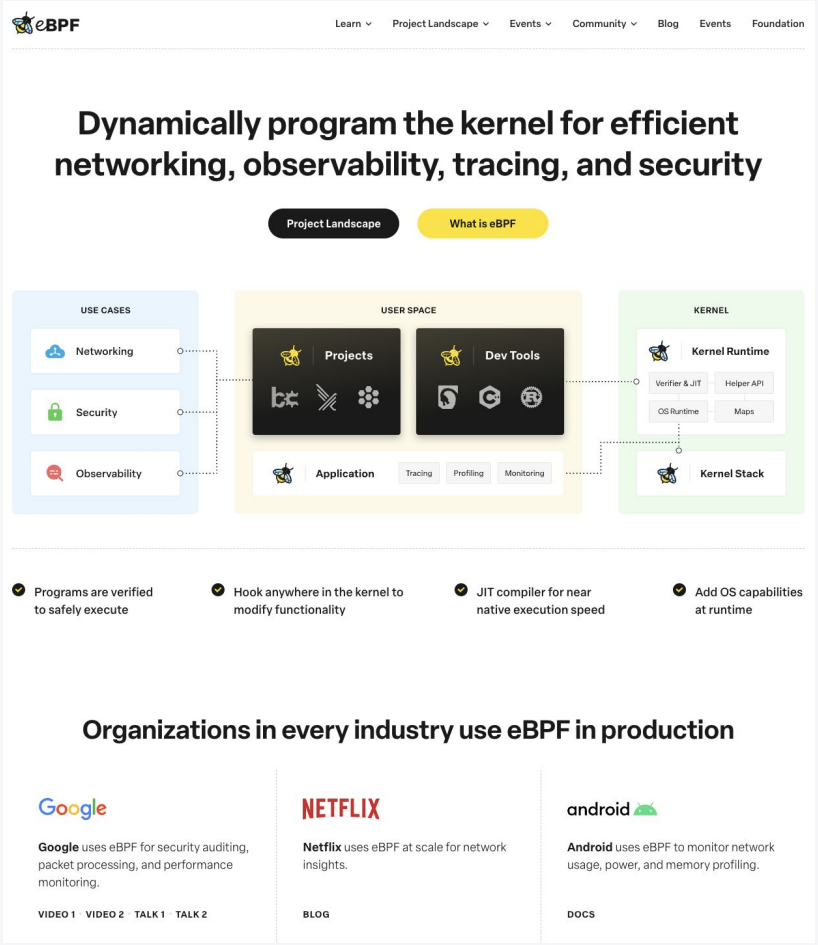
History [edit]

Evolution from classic BPF [edit]

eBPF was built on top of the Berkeley Packet Filter (cBPF). At the lowest level, it introduced the use of ten 64-bit registers (instead of two 32-bit long registers for cBPF), different jump semantics, a call instruction and corresponding register passing convention, new instructions, and a different encoding for these instructions.[13] A number of additional features were subsequently added. The evolution of eBPF took many years and a large community of contributors, and is still ongoing. The table below summarizes some of the most significant milestones of this evolution:

ISOVALENT

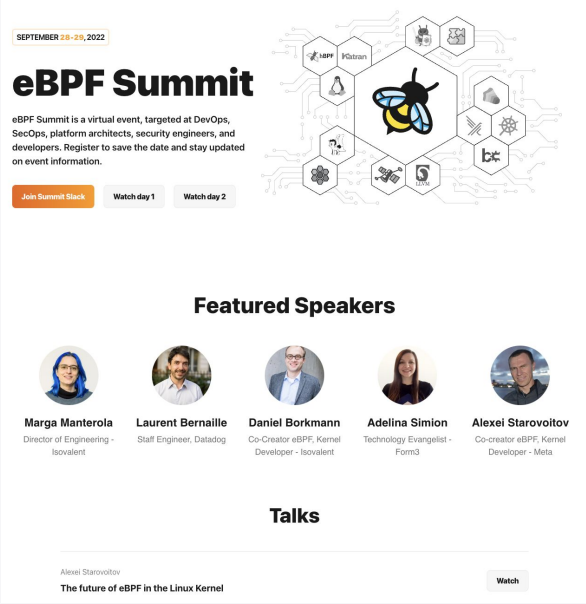
ebpf.io

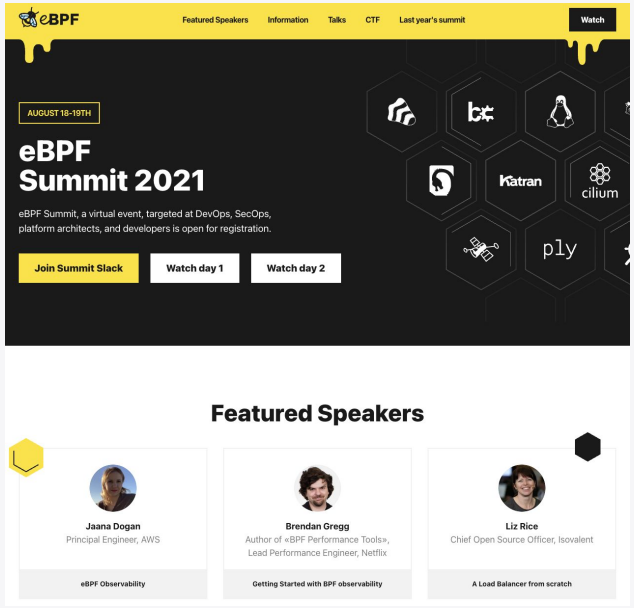


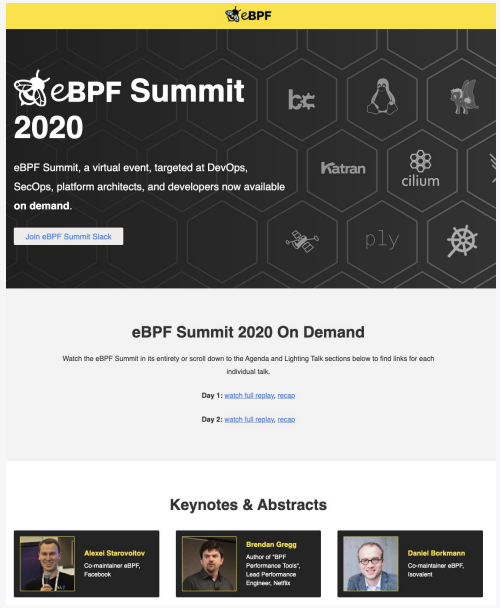
@breakawaybilly

ISOVALENT

eBPF Summit





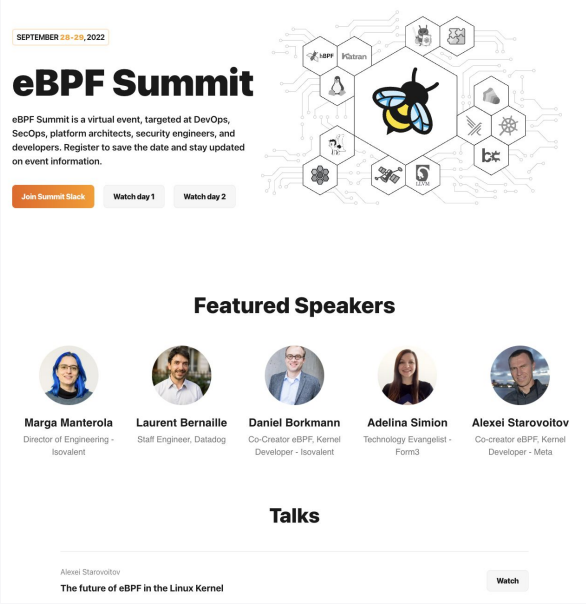


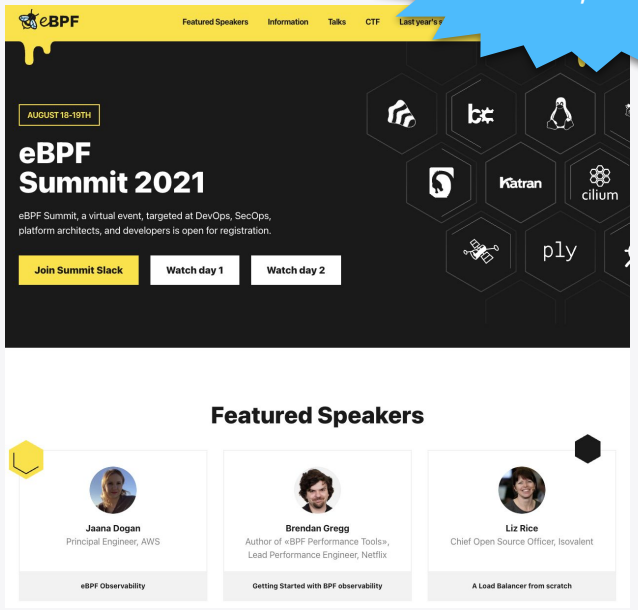
@breakawaybilly

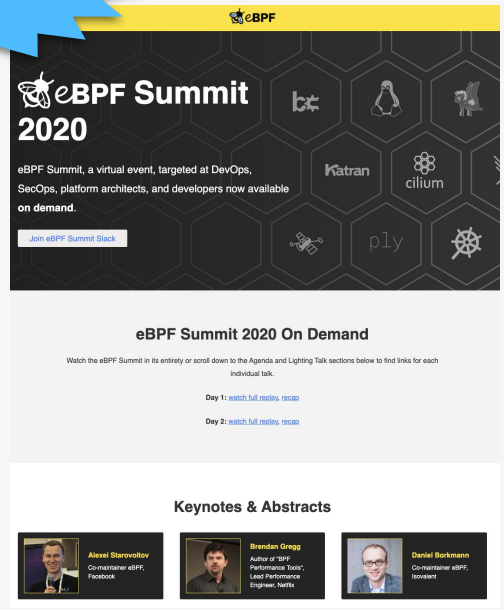
ISOVALENT

eBPF Summit

September 13th, 2023







@breakawaybilly

ISOVALENT

A Vision for eBPF Inside

ISOVALENT

A Vision for eBPF Inside

@breakawaybilly

ISOVALENT

eBPF is the Cloud Native App Store

@breakawaybilly

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

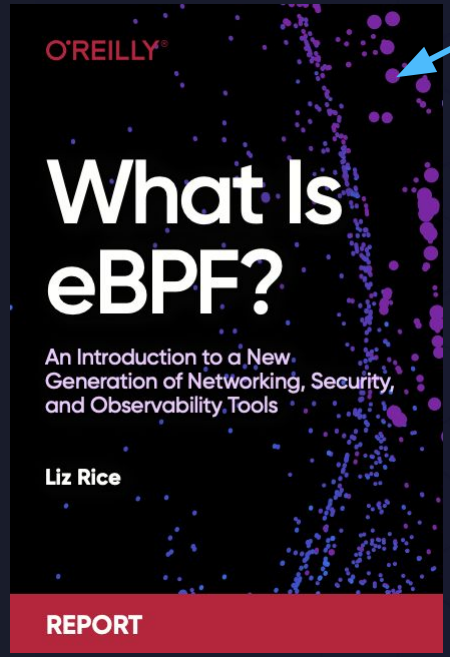
ISOVALENT

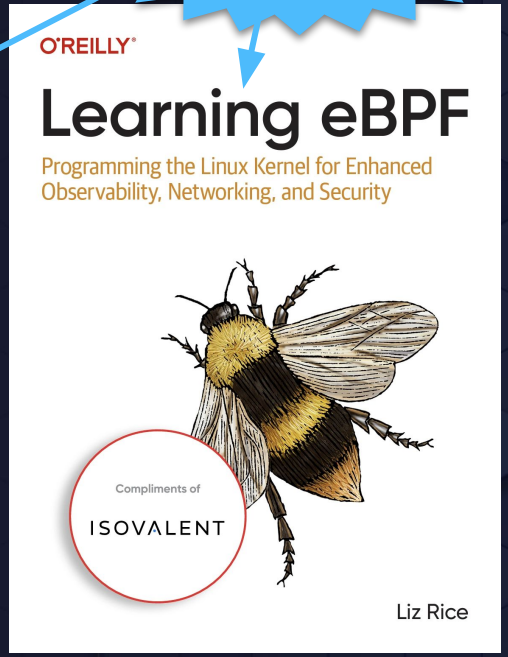
Thank you

ebpf.io

@breakawaybilly

ebpf.io





Download from **isovalent.com**