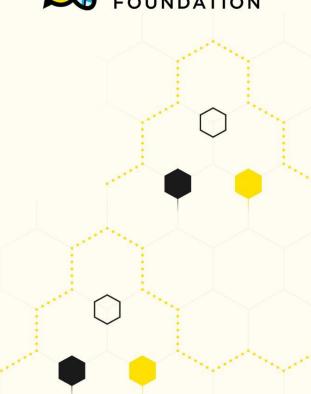


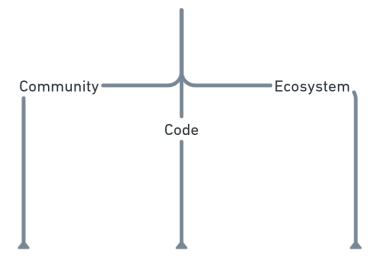
From Projects to Ecosystems: Lessons from the eBPF Foundation



# To innovate and enable others to innovate

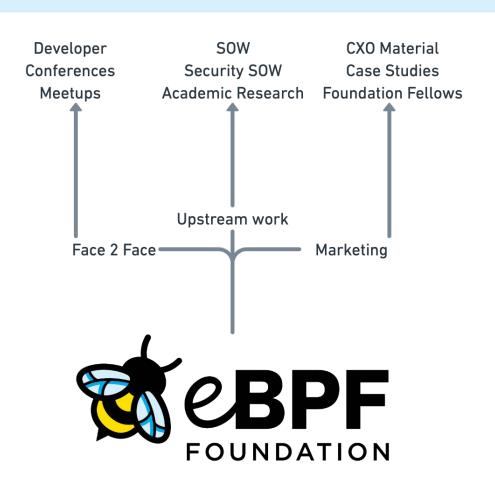






# eBPF Foundation Strategy





# Developer Face to Face





### Linux Kernel Developers' bpfconf 2025



bpfconf is an invitation-only technical workshop run by the Linux community in order to bring BPF core developers together, to discuss new ideas and to work out improvements to the BPF subsystem that will make their way into future mainline kernels and into the BPF compiler backends

The conference is purposely kept small with focus on discussion rather than just presentation. Along with the LPC's <u>BPFTrack</u> which is organized and run by the same community, the groat is to allow developers to meet face to face twice per year to exchange and discuss ongoing developments in the BPF ecosystem.

The 2025 befoon edition is a three-days conference which is part of the  $\underline{LSF/MM/BPF}$  summit. It is therefore also open to all LSF/MM/BPF attendees.

### Discussion Topics

The following discussion topics have been brought up at this year's bpfconf. In each slot below, there is a short discussion topic with a link to the corresponding slides in case slides have been used as a discussion starter.

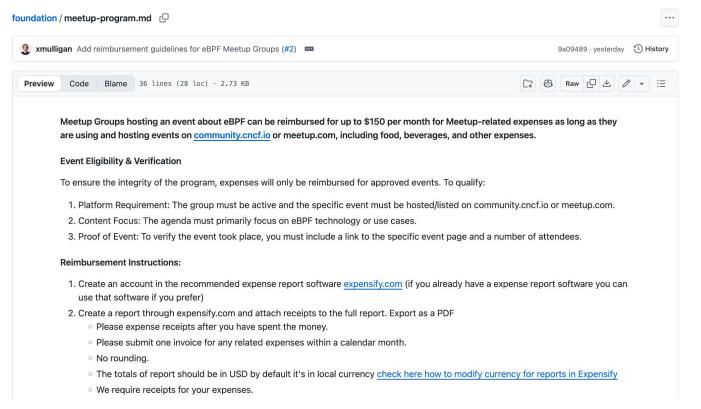
LWN coverage of the BPF track from Jonathan Corbet can be found  $\underline{\text{here}}.$ 





# Developer Face to Face





\* Meetup program coming soon

# Upstream Development **SOW**





# index: kernel/git/torvalds/linux.git

Linux kernel source tree

master ~ switch

Linus Torval

out summary	refs log tree commit diff stats			log msg V eBPF Fou s
Age	Commit message (Expand)	Author	Files	Lines
days	Merge branch 'selftests-bpf-convert-test_tc_edt-sh-into-test_progs'	Alexei Starc	riles	-107/+151
days days	selftests/bpf: do not hardcode target rate in test_tc_edt BPF program	Alexis Lotho é (eBPF Foundation)	2	-3/+4
days days	selflests/bpf: remove test_tc_edt.sh	Alexis Lotho (eBPF Foundation)	2	-102/+0
,		Alexis Lotho (eBPF Foundation)	2	-1/+145
3 days	selftests/bpf: integrate test_tc_edt into test_progs		2	
days	selftests/bpf: rename test_tc_edt.bpf.c section to expose program type	Alexis Lotho (eBPF Foundation)	2	-2/+3
2025-11-06	selftests/bpf: Use start_server_str rather than start_reuseport_server in tc	Alexis Lotho é (eBPF Foundation)	1	-15/+13
2025-11-06	selftests/bpf: Systematically add SO_REUSEADDR in start_server_addr	Alexis Lotho é (eBPF Foundation)	1	-1/+7
2025-10-31	selftests/bpf: Add checks in tc_tunnel when entering net namespaces	Alexis Lotho (eBPF Foundation)	1	-46/+88
2025-10-31	selftests/bpf: Skip tc_tunnel subtest if its setup fails	Alexis Lotho é (eBPF Foundation)	1	<del>-2</del> /+2
025-10-31	Merge branch 'selftests-bpf-integrate-test_xsk-c-to-test_progs-framework'	Alexei Starc oitov	6	-2632/+3092
025-10-31	selftests/bpf: test_xsk: Integrate test_xsk.c to test_progs framework	Bastien Curetchet (eBPF Foundation)	5	-3/+163
025-10-31	selftests/bpf: test_xsk: Isolate non-Cl tests	Bastien Curatchet (eBPF Foundation)	2	-17/+32
025-10-31	selftests/bpf: test_xsk: Don't exit immediately on allocation failures	Bastien Curatchet (eBPF Foundation)	3	-42/+110
025-10-31	selftests/bpf: test_xsk: Don't exit immediately if validate_traffic fails	Bastien Curatchet (eBPF Foundation)	1	-4/+8
025-10-31	selftests/bpf: test_xsk: Don't exit immediately when workers fail	Bastien Curetchet (eBPF Foundation)	1	-34/+76
025-10-31	selftests/bpf: test_xsk: Don't exit immediately when gettimeofday fails	Bastien Curatchet (eBPF Foundation)	1	-4/+4
025-10-31	selftests/bpf: test_xsk: Don't exit immediately when xsk_attach fails	Bastien Curetchet (eBPF Foundation)	1	-9/+19
025-10-31	selftests/bpf; test_xsk; Add return value to init_iface()	Bastien Curatchet (eBPF Foundation)	3	-6/+11
025-10-31	selftests/bpf: test_xsk: Release resources when swap fails	Bastien Cur tchet (eBPF Foundation)	1	-1/+6
025-10-31	selftests/bpf: test_xsk: Wrap test clean-up in functions	Bastien Cur tchet (eBPF Foundation)	1	-12/+24
025-10-31	selftests/bpf: test xsk: fix memory leak in testapp xdp shared umem()	Bastien Cur tchet (eBPF Foundation)	1	-1/+22
025-10-31	selftests/bpf: test_xsk: fix memory leak in testapp_stats_rx_dropped()	Bastien Cur tchet (eBPF Foundation)	1	-0/+7
025-10-31	selftests/bpf: test_xsk: Fixtestapp_validate_traffic()'s return value	Bastien Cur tchet (eBPF Foundation)	1	-1/+4
025-10-31	selftests/bpf: test_xsk; Initialize bitmap before use	Bastien Cur tchet (eBPF Foundation)	1	-0/+4
025-10-31	selftests/ppf: test_xsk; Split xskxceiver	Bastien Cur tchet (eBPF Foundation)	5	-2634/+2738
025-10-29	selftests/ppf: Remove test to tunnel.sh	Alexis Lotho (GBPF Foundation)	2	-321/+0
025-10-29	selftests/bpf: Integrate test_tc_tunnel.sh tests into test_progs	Alexis Lotho é (eBPF Foundation)	2	-19/+693
025-10-29	selftests/bpf: Make test_tc_tunnel.bpf.c compatible with big endian platforms	Alexis Lotho (eBPF Foundation)	1	-35/+22
025-10-29	selfests/bpf: Add to helpers	Alexis Lotho (eBPF Foundation)	2	-93/+74

# **Upstream Development**





# index: kernel/git/torvalds/linux.git

about summary refs log tree commit diff stats

Alexis Lothoré (eBPF Foundation) <alexis.lothore@bootlin.com> 2025-11-28 23:27:19 +0100 author committer Alexei Starovoitov <ast@kernel.org> 2025-11-29 09:37:41 -0800

b0f82e7ab6fb2f8501ef87ae928cbf7358d7845e (patch) commit

b349f81a8b8e95c9b5241b8e771c14e774222e36

4b4833acc63e9c8ea9d5897ee84b694f30b23882 (diff)

linux-b0f82e7ab6fb2f8501ef87ae928cbf7358d7845e.tar.gz

### selftests/bpf; integrate test to edt into test progs

test\_tc\_edt.sh uses a pair of veth and a BPF program attached to the TX veth to shape the traffic to 5MBps. It then checks that the amount of received bytes (at interface level), compared to the TX duration, indeed matches 5Mbps.

Convert this test script to the test\_progs framework:

- keep the double veth setup, isolated in two veths
- run a small tcp server, and connect client to server
- push a pre-configured amount of bytes, and measure how much time has been needed to push those
- ensure that this rate is in a 2% error margin around the target rate

This two percent value, while being tight, is hopefully large enough to not make the test too flaky in CI, while also turning it into a small example of BPF-based shaping.

Signed-off-by: Alexis Lothoré (eBPF Foundation) <alexis.lothore@bootlin.com> Link: https://lore.kernel.org/r/20251128-tc edt-v2-2-26db48373e73@bootlin.com Signed-off-by: Alexei Starovoitov <ast@kernel.org>

-rw-r--r-- tools/testing/selftests/bpf/prog\_tests/test\_tc\_edt.c 144 -rw-r--r- tools/testing/selftests/bpf/progs/test\_tc\_edt.c

2 files changed, 145 insertions, 1 deletions

diff --git a/tools/testing/selftests/bpf/prog tests/test to edt.c b/tools/testing/selftests/bpf/prog tests/test to edt.c new file mode 100644

index 00000000000000..9ba69398eec49c

- --- /dev/null
- +++ b/tools/testing/selftests/bpf/prog\_tests/test\_tc\_edt.c

@@ -0.0 +1.144 @@



Engineering Training

# Improving the eBPF tests in the kernel



Alexis Lothoré March 17, 2025 Technical

ebpf, testing, upstream

As part of a partnership with the eBPF Foundation, Bootlin engineers Bastien Curutchet and Alexis Lothoré are working with the kernel community in order to improve eBPF support in the kernel on different aspects. This post is the first one of a series highlighting this effort. For those who need to catch up with the eBPF technology, you can take a look at our "Linux Debugging, tracing



and profiling" training course which has been recently updated with eBPF basics!

# Upstream Development - BTF, Trampolines, and CI



- Runtime Type Information via BTF: Solved DWARF size constraints (reducing ~200MB to ~5MB) allowing embedding runtime-loadable type data for dynamic function modeling
- Low-Overhead Trampolines: Replaced expensive kprobes with direct function calls acting as a dynamic ABI bridge for fentry/fexit
- **ARM64 Feature Parity**: Multi-kprobes and complex argument handling for functions with >8 parameters (stack-passed args)
- Test Suite Modernization: Migrated legacy standalone shell scripts to the generic test\_progs C runner
- Expanded CI Coverage: Increased automated regression testing reliability on bpf-next

# What SOWs should we fund in the future?



- riscv64 BPF JIT on par with x86-64 & arm64
- XXX

# **Upstream Security Funding**



# nccgroup

### eBPF Verifier Code Review

NCC Group Version 1.0 – November 11, 2024



Prepared By Chris Anley Nathaniel Theis Prepared For eBPF Foundation

### 5 Finding Details

### find\_equal\_scalars Mishandles 32-Bit Addition

 Overalt Risk
 High
 Finding ID
 NCC-E015561-JJX

 Impact
 High
 Component
 eBF verifier

 Exploitability
 Low
 Category
 Data Validation

 Status
 Fixed

#### Impact

An attacker with CAP\_BPF (required to reach the vulnerable code paths) who can load & execute eBPF programs on a vulnerable system can read to, and write from, arbitrary kernel memory.

#### Description

A vulnerability in the eBPF verifier permits an attacker to submit and run an eBPF program that can read from, and write to, arbitrary kernel memory.

The issue involves the tracking of eBPF register values, and specifically the identification of equal scalars. The specific vulnerable code is related to the <code>find\_equal\_scalars()</code> function and the <code>BPF\_ADD\_CONST</code> flag, which signifies a constant offset between two scalar registers.

An attacker requires either root privilege or CAP\_BFF<sup>2</sup> to successfully exploit the issue. Additionally, the POC code below requires CAP\_PERFMON, because it doesn't bypass the ALU santlizer (although in an actual exploit this can be achieved with previously documented<sup>3</sup> methods).

The verifier attempts to track "similar" scalars in order to propagate bounds information learned about one scalar to others. For instance, if r1 and r2 are known to contain the same value, then upon encountering if r(r1 = %1234) goto 1234, not only does it know that r1 is equal to 0x1234 on the path where that conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where that conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where that conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where that conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the conditional jump is not taken, it also knows that r2 is equal to 0x1234 on the path where the 0x1234 on the 0x1234 o

Additionally, if env->bpf\_capable (i.e. if the process loading this eBPF program has CAP\_BPF), the verifier will track scalars which should be a constant delta apart (if r1 is known to be one greater than r2, then if r1 is known to be equal to 0x1234, r2 must be equal to 0x1233.)

The relevant code from adjust\_reg\_min\_max\_vals(), located at kernel/bpf/verifier.c:14101:

2. CAP\_BPF is a Linux kernel capability which serves several purposes. First, it allows certain eBPF operations such as creating maps. Second, it enables more sophisticated analysis in the eBPF verifier (including the vulnerable scalar-offset analysis documented in this finding). Third, if unprivileged eBPF is disabled by the unprivileged\_bpf\_disabled syscit—as it currently is on most popular distributions by default: CAP\_BPF is required to load any BPF program.

 A technique is described in this writeup by Manfred Paul of his winning entry in the Pwn2Own 2020 competition.

7 / 31 - Finding Details





# **Upstream Security Funding**



# \$225,000 grant from Alpha-Omega for:

- Defensive runtime tooling
  - Enabling KASAN for JIT-compiled eBPF programs to validate memory accesses
- Auditing
  - Security review of x86-64, arm64, and riscv64 JIT compilers, specifically targeting vulnerabilities in instruction encoding, register allocation, and immediate value handling
  - Check verifier-to-JIT integrity ensuring security assumptions (memory boundaries, control flow) are preserved in translation
  - Assessment of unprivileged surfaces, cBPF to eBPF translation (bpf\_convert\_filter) and seccomp filter validation

# How can we improve eBPF security?





# Academic Research Grants



Announcements Blog

# eBPF Foundation Announces \$250,000 in Grant Awards for Five eBPF Academic Research Projects

By Dan Brown August 29, 2024 7 min read

Announcements Blog

Projects will advance eBPF's open source technology by improving scalability, static an and more

# eBPF Foundation Awards \$100,000 in Research Grants to Advance eBPF Safety and Efficiency

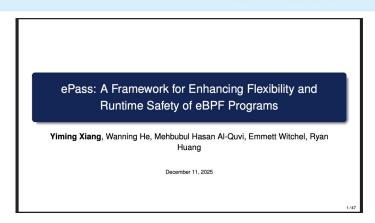
By Dan Brown September 16, 2025 4 min read

Research projects will strengthen eBPF programmability, runtime safety, and datacenter energy efficiency through verifier-cooperative instrumentation and QoS-aware power management

# Academic Research Grants



- Enhancing Flexibility and Safety of eBPF Programs through Verifier-Cooperative Instrumentation
- eBPF Governors: QoS-aware Agile Power Management



- Efficient IO-Intensive µs-scale Applications using eBPF
- Verified Path Exploration for eBPF Verifier Static Analysis
- Lazy Abstraction Refinement with Proof for an Enhanced Verifier
- Improving eBPF Complexity with a Hardware-backed Isolation Environment
- Learned Virtual Memory with eBPF

# What research is important?





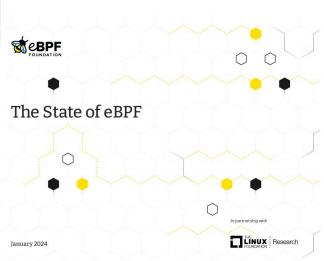
# **CXO Education Material**





# eBPF: Unlocking the Kernel [OFFICIAL DOCUMENTARY]

150K views • 1 year ago





# Case Studies - Your company next?



### BYTEDANCE USES EBPF TO **ENHANCE NETWORKING** PERFORMANCE

#### OVERVIEW

Bytedance, a global technology company operating a wide range of dyseance, a grount sechnology company operating a wide range or content platforms around the world at massive scale, faced significant challenges in ensuring performance and stability across handle high throughput while maintaining stability. By leveraging

#### CHALLENGE

- Performance Bottlenecks: The virtual Ethernet-based solution introduced soft-interrupt bottlenecks in the network stack causing inefficiencies
- was critical, but unproven solutions posed risks in production . Kernel Version Constraints: Upgrading to the required kerne
- version (6.8) was not immediately feasible due to operational To address these challenges. Bytedance required a robust

scatable, and high-performance solution that could be deployed incrementally across its data centers.

### SOLUTION

Bytedance turned to eBPF to address these challenges, eBPF, a powerful technology for dynamically and safely reprogramming the Linux kernel, enabled the company to redesign its containe etworking stack. Key steps in the implementation included: · Introducing netkit:

- Bytedance adopted <u>netkit</u> an eBPF-powered networking device introduced in version 6.8 of the Linux kernel, which
- To accommodate existing constraints and legacy denloyments netkit was backported to kernel version Rolling Upgrade Strategy:

### minimal disruption. Both the container networking

- nterface and kernel upgrades were managed dependently, allowing a gradual transition to netkit. There was a concern over potential issues with mixed deployments, so compatibility between netkit and virtual
- Ethernet was ensured throughout the transition
- scenarios where netkit or its associated eBPF programs



### The deployment of eBPF and netkit delivered significant

- improvements across Bytedance's data centers: - Performance Gains:
- Resolved issues with high CPU load and packet reordering caused by virtual Ethernet, enhancing overall
- Scalability and Stability: Successfully deployed netkit across dozens of clusters demonstrating its reliability and readiness for broader
- Operational Benefits: Simplified the networking stack, reducing maintenance overhead and improving system observabilit

### FUTURE PLANS

· Hardware Offloading: Combining . Broader Use Cases: Expanding el

### container networking to other ar

CONCLUSION REDUCE CPU CYCLES AND

scale networking challenges. By rede and leveraging the flexibility of eBPF. SERVER DEMANDS BY UP and scalability. This deployment servi TO 20% To learn more about their use of netk OVERVIEW

Meta, a global leader in technology and social networking, faced the challenge of providing comprehensive profiling data for its backend services without disrupting performance. To address this. Meta developed <u>Strobelight</u>, a profiling orchestrator that leverages eBPF to collect observability data efficiently. This solution has driven easurable efficiency improvements across Meta's infrastructure resulting in substantial capacity savings and operational benefits. CHALLENGE

Meta needed a way to gather and normalize profiling data across its vast and varied backend services without introd overhead that could impact performance. The key challenges

Ensuring minimal disruption to live services while collecting

META'S STROBELIGHT

LEVERAGES EBPF TO

- Making profiting data uniform and easily interpretable Preventing system overloads due to excessive data storage
- · Supporting multiple kernel versions across Meta's

#### SOLUTION

To overcome these challenges. Meta implemented Strobelight, a ophisticated profiling orchestrator that integrates multiple ofiling tools, including eBPF. Strobelight enables engineers to CBI I time spent in function calls and execution paths

· Al/GPU profiling and memory tracking.

overhead data collection, avoiding additional instrumentation inside binaries and maintaining efficient performance. Additionally, given Meta's diverse infrastructure with multiple

kernel versions. Strobelight was designed to ensure: Feature compatibility across different kernel versions, with appropriate fallbacks. Dynamic sampling to balance data collection rates and

Concurrency and queuing safeguards to prevent.

performance degradation

SOFTWARE

### auditing, and real-time behavioral enforcement. AntCWPP has Meta LLENGES al technology leader. Ant Group operates diverse

The deployment of eBPF within Strobelight has led to significant

- single one-character code change.

   20% reduction in CPU cycles, equating to a 10-20% reduction in the number of required servers for Meta's top services.
- Faster debugging and performance analysis, allowing engineers to prevent regressions before they reach
- Dynamic sampling mechanisms, optimizing profiling rates without overloading storage systems.

### WHY FRPF?

eBPF was selected as the core technology due to its low helpers; and lack of requirements for additional instrumentation.

nstrumentation inside binaries and impact runtime performance. Strobelight's eBPF-based approach enabled real-time profiling without modifying application code, broader observability across multiple languages and systems, and efficient data collection with minimal overhead.

#### NEXT STEPS

- Meta continues to expand its use of eRPF to further enhance AL/ML workloads
- More complex efficiency analyses for improved resource
- Open sourcing Strobelight's profilers and libraries for broader adoption within the open source community.

# Configerator Strobelight Strobelight Strobelisht

#### SOLUTION

Ant Group built AntCWPP, a next-generation workload prote platform, by combining Kata Containers and eBPF

 Kata Containers for Isolation Each Kata container runs inside a tightweight VM with its own

GROUP

ANT GROUP SECURES THEIR

CONTAINERS AND eBPF FOR

Ant Group needed to strengthen security for its large scale

containerized workloads, which power their critical online service

solutions left gaps around container escape, inconsistent kernel

Platform (AntCWPP), combining Kata containers with eBPF. This

red workloads across massive production clusters.

ses (e.g., runc), containers share the host kernel. This

ing isolation, and a single compromise can affect

se security needs: Internet-facing services, internal

sistent kernel versions: Large-scale production

ecture makes it difficult to fully prevent attackers from

ations, and Al agents require different security policies

at CWPP solutions built on kernel modules or host-leve

Contining the eBPF features

ring the need for an approach that could combine

station with flexible, fine-grained enforcement.

FINE GRAINED CONTROL

PLATFORM WITH KATA

OVERVIEW

sie workloads.

- dedicated kernel, preventing workloads from impacting each other or the host. This design eliminates container escape and reduces the blast radius of attacks. eBPF for Auditing and Enforcement
- · AntCWPP attaches eBPF programs to Linux Security Module (LSM) hooks and network control points inside the Kata VM
- Process execution monitoring and drift prevention
- Network activity auditing and fine-grained enforcement
   File integrity monitoring (FIM) and path-based access control System call monitoring and real-time attack interception
- Security Agent for Policy Management A node-level security agent delivers policies to Kata Pods. load. eRPF neograms into VM kernels, and collects logs through eBPF the host kernel, enabling flexible and application specific

By deninging AntCWPD across bioburisk workloads. Ant Group

· Reduced container escape: Kata's independent kernel. significantly reduced escape risks. Application-level security control: eBPF-based policies allow



### **Rakuten** Mobile RAKUTEN MOBILE ADOPTS FRP

TO STRENGTHEN ANOMALY **DETECTION AND SECURITY IN CLOUD-NATIVE TELECOM** 

### OVERVIEW

As mobile networks transition to virtualized and cloud-native erchitectures, traditional monitoring tools struggle to keep page. To iddress this challenge, Rakuten Mobile adopted open source eBPF technology, originally designed for cloud and enterprise

observability, and adapted it to the complex demands of telecom networks. The solution, Sauron eBPF, enables real-time telemetry, anomaly detection, security tracing, and observability across

### CHALLENGES

- Overwhelming Alert Volume: Static rules and threshold triggered large volumes of false positives, exhausting operational resources and reducing system trust.
- . Poor Adaptability: Legacy solutions lack the flexibility to

#### SOLUTION

Rakuten Mobile developed Sauron eBPF, a lightweight, flexible security gaps in telecom networks. It provides:

Network Performance (Transport): Active hop-by-hop testing and bottleneck prediction with eBPF. Real-time telemetry on







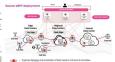






aying the foundation for intelligent, self-healing 5G and 6G

To learn more about their use of Sauron eBPF, check out the article



### CBPF (-) Alibaba Cloud · Scalable Metric Design: Lightweight metrics allow real-time updates per I/O event with peoliptic creekend.

ALIBABA CLOUD LEVERAGES eBPF FOR ADAPTIVE LAYER 7 LOAD BALANCING

#### OVERVIEW

Alibaba Cloud is one of the largest public cloud providers, operating event notification mechanisms struggled to keep up. To solve this userspace-directed I/O event notification framework powered by eBPF. By integrating real-time feedback from application workers greater infrastructure efficiency. This eBPF-driven approach strengthens the stability and scalability of Alibaba Cloud's global network infrastructure, reducing operational overhead, lowering

Alibaba Cloud's L7 load balancers must sustain biob throughout

time insights.

SOAR platforms.

- while ensuring fair traffic distribution across multiple workers Existing Linux mechanisms presented several limitations:

  • Unbalanced Workloads: epoll's LIFO wakeup behavior
- introduced to mitigate the thundering herd problem, caused · Observability (Traffic Mire Native Network Function
- counts, preventing adaptive scheduling.

  Operational Risk: Modifying the kernet was not feasible at scale, where even minor bugs could cause large-scale service. policy-driven alerts on the Enhanced AL/ML Capabili Performance Sensitivity: Maintaining low latency under multi-
- time threat detection fron Tenant Performance Isolation: Load-balancing mechanism failed to evenly distribute resources arms tenants, causing
- Hop-by-Hop Visibility: § safety extend temperature intelligence intelligence intelligence intelligence.

  Hop-by-Hop Visibility: § safety extend termed scheduling behavior white incorporating performance enabled a dynamic, userspace-driven intelligence.

. Broader eBPF Use Case . Lock-Free Synchronization: Worker status is exchanged through

### Reduced False Positive allowing teams to focus SOLUTION

- automated monitoring.

  Improved Threat Detec Alibaba Cloud created Hermes, a userspace-directed I/O er notification framework that leverance eRPF to aurement the Line
- respond to previously u
  including zero-day thre
  Increased Efficiency: Di
  conceins deficiency Di
  conceins deficiency Di
  connection dispatch decisions reflect actual worklead conditions: and intervention reduce 

  Closed-Loop Scheduling Userspace workers continuously

- control plane behavior : control plane behavior : defice operations on ellPf array maps, enabling high-speed updates without tooks or contention.
- · Cross-Domain Correlation: Linking telemetry from multiple
- domains (compute, storage, network) for full-stack observability.

telemetry with Al-driven analytics. By bridging the gap between



### updates per I/O event with negligible overhead. Two-Stage Load Distribution: Userspace preselects eligi

workers, and the kernel refines selection using a hash of the Production-Safe Implementation: The kernel dispatch logic

> and is production-ready at scale. This architecture enables Hermes to adapt continuously to runtime conditions while preserving system stability and efficiency



### RESULTS

Hermes has delivered measurable and sustained improvement

- 99.8% reduction in daily worker hangs, as measured by health
- probes exceeding 200ms latency.

  Consistent performance across diverse traffic models, with best
- or near-best latency and throughput in all cases. Proven scalability, deployed on 100,000 CPU cores and



DATADOG USES EBPF TO

OBSERVABILITY ACCURACY

ormed these actions by relying on Notlink and container ru

Network Address Translation: Originally, CNM acquires

processor or some connections.

Tracking Process Launches: The Datadog Agent was also using Nettink updates for process launches to hook upon

IMPROVE NETWORK

AND PERFORMANCE

CHALLENGE

# DATADOG

### SOLUTION

Datadog used eBPF to come up with a couple of appro eBPF-based Connection Tracker Using eBPF <u>sprobes</u>, the team at Data

#### RESULTS

eBPF-based Connection Tracker

Process Event Data Stream

Real-time updates from eBPF also incurred a lower perfor penalty for attaching uprobes compared to Netlink.

### NEXT STEPS



# ByteDance's Container





VI Content Podosot Articles web3 Cloud Engineering Business & Philosophy Data Open Source Machine L

SOFTWARE ENGINEERING DAILY

# Community Building Fellowships



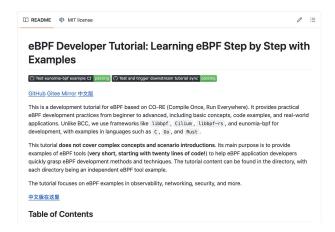
Uncategorized

# eBPF Foundation Announces Inaugural Community & Advocacy Fellowship Recipients

By Dan Brown October 15, 2025 4 min read

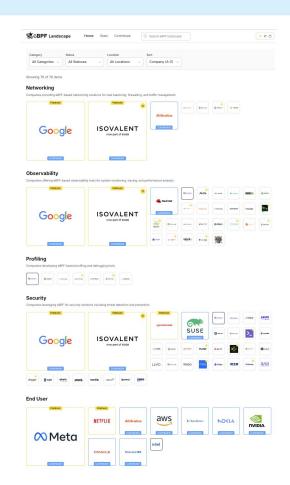
# TL;DR

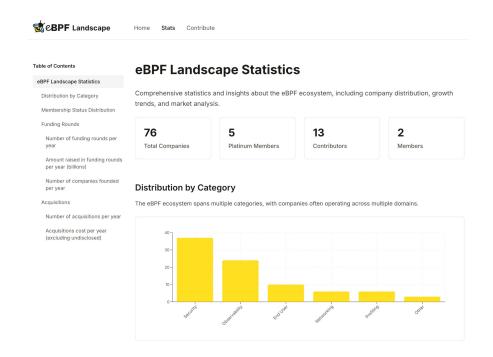
The eBPF Foundation announced the recipients of its inaugural Community & Advocacy Fellowship, recognizing contributors expanding global access to eBPF through education, tutorials, and community-building. The 2025 fellows are Teodor Janez Podobnik (eBPFChirp Newsletter) and Yusheng Zheng (bpf-developer-tutorial).



# Company Landscape







# What information would be helpful to surface?





# Join the Foundation



# Who are our members

**Platinum** 











Silver





**TOYOTA** 

# Join the Foundation



# How you benefit from membership

### Strategic Influence

Take part in shaping the technical roadmap and key Work openly with leading industry decisions for the future of eBPF.

### Security & Research

Contribute to security audits, threat modeling, and academic research to strengthen eBPF.

### Cross-Platform Collaboration

leaders and developers in a neutral, collaborative environment.

### Visibility & Credibility

Build your organization's reputation as a trusted and visible leader in the open source ecosystem.

### Flexible membership tiers for every organization

Choose the level that aligns with your organization's size and goals. All tiers include participation, visibility, and collaboration opportunities.

### Silver

### \$5,000 - \$30,000

- 5,000+ Employees \$30,000 (\$10,000 For LF Members)
- 500-4,999 Employees \$25,000 (\$10,000 For LF Members)
- 100-499 Employees \$15,000 (\$5,000 For LF Members)
- Up to 99 Employees \$10,000 (\$5,000 For LF Members)

### Get started

Shared influence, training perks, marketing access, and brand

- Possible Board seat (1 per 5 Silver members)
- 10 free training tickets
- Participate in marketing & community
- Logo displayed on website & materials
- Linux Foundation Member Summit access

#### Platinum

### \$70,000

• \$50,000 if already an LF Member



Voting rights, top visibility, leadership access, and training perks.

- ✓ 1 voting rep to Board, Marketing & subcommittees
- Premium member visibility
- Access meetings & events
- 10 free training tickets
- Quarterly membership announcement
- Direct leadership insights
- Linux Foundation Member Summit invite
- Conference & sponsorship discounts

# Get in Contact



Email me: bill@ebpf.io

Email BSC: info@ebpf.foundation

eBPF Japan Community <a href="https://ebpf.connpass.com/">https://ebpf.connpass.com/</a>

# Discussion

