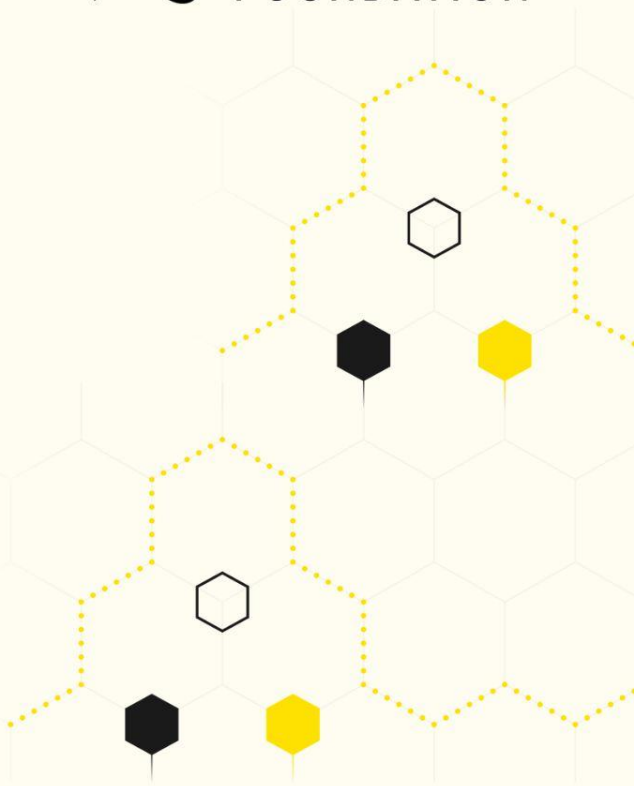
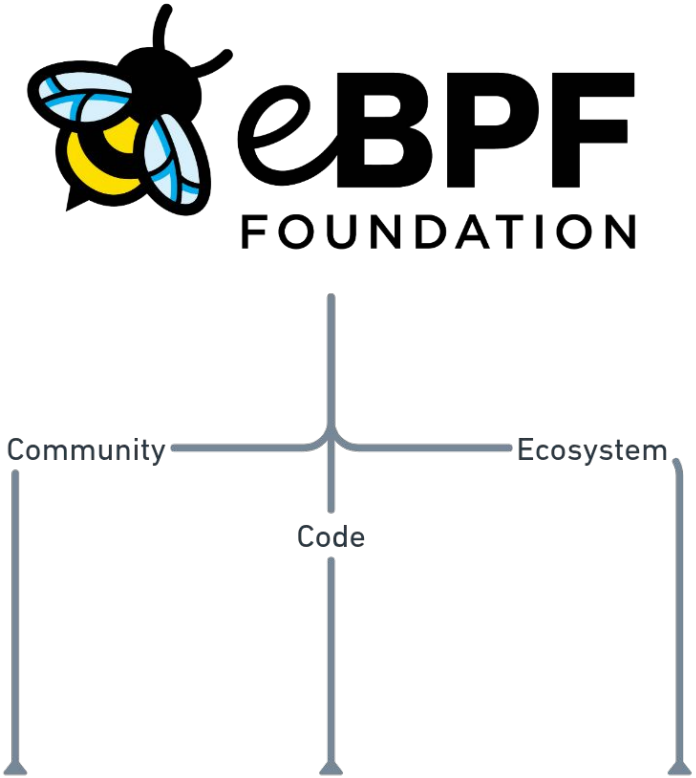
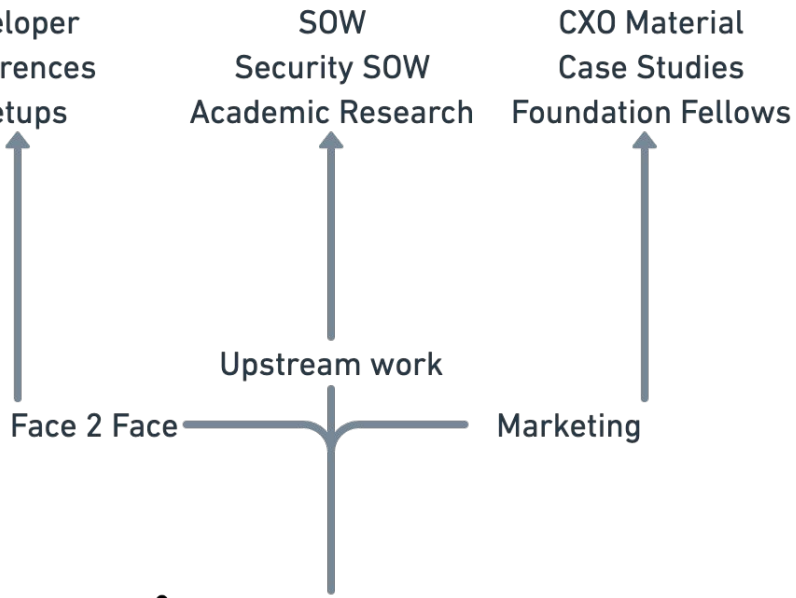
From Projects to Ecosystems: Lessons from the eBPF Foundation



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 BPF Track which is organized and run by the same community, the goal is to allow developers to meet face to face twice per year to exchange and discuss ongoing developments in the BPF ecosystem.

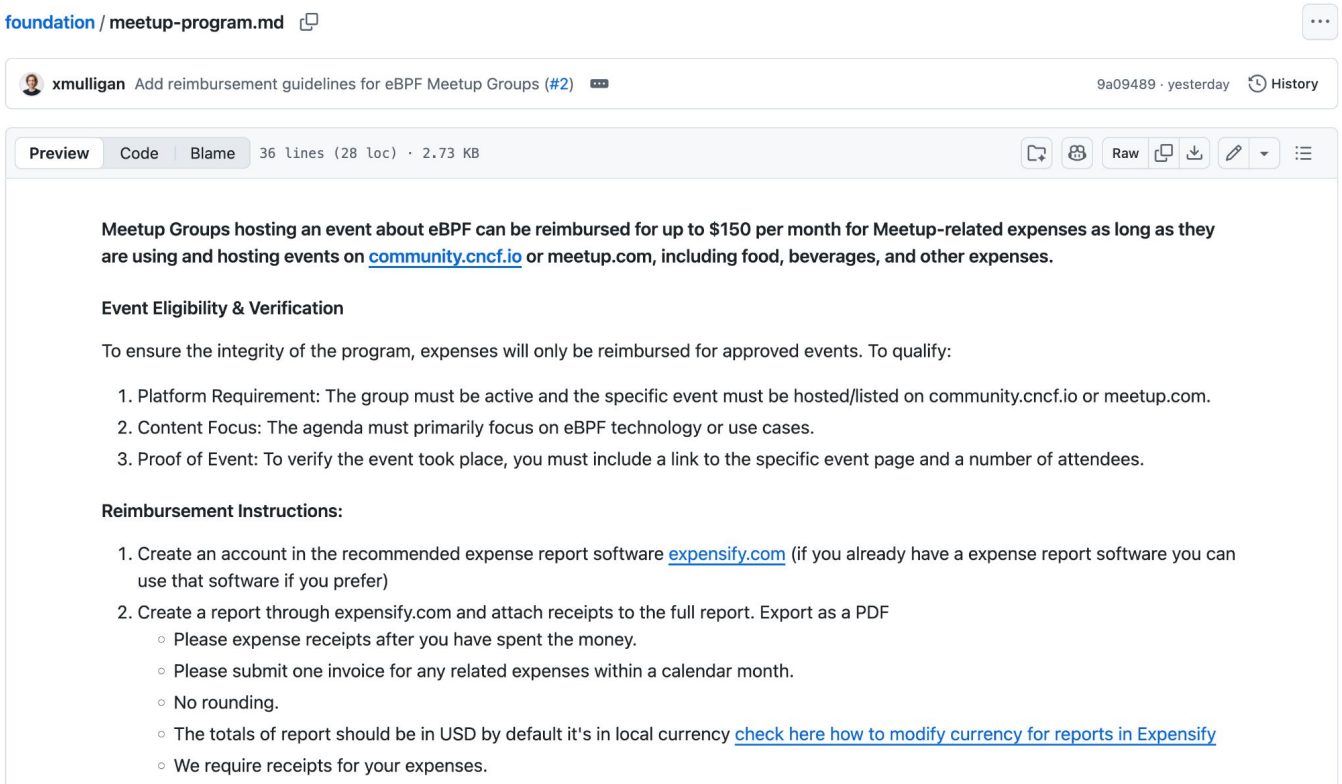
The 2025 bpfconf edition is a three-days conference which is part of the LSF/MM/BPF summit. It is therefore also open to all LSF/MM/BPF attendees.

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 here.

Developer Face to Face



foundation / meetup-program.md

xmulligan Add reimbursement guidelines for eBPF Meetup Groups (#2) 9a09489 · yesterday History

Preview Code Blame 36 lines (28 loc) · 2.73 KB Raw

Meetup Groups hosting an event about eBPF can be reimbursed for up to $150 per month for Meetup-related expenses as long as they are using and hosting events on community.cncf.io or meetup.com, including food, beverages, and other expenses.

Event Eligibility & Verification

To ensure the integrity of the program, expenses will only be reimbursed for approved events. To qualify:

1. Platform Requirement: The group must be active and the specific event must be hosted/listed on community.cncf.io or meetup.com.
2. Content Focus: The agenda must primarily focus on eBPF technology or use cases.
3. Proof of Event: To verify the event took place, you must include a link to the specific event page and a number of attendees.

Reimbursement Instructions:

1. Create an account in the recommended expense report software expensify.com (if you already have a expense report software you can use that software if you prefer)
2. Create a report through expensify.com and attach receipts to the full report. Export as a PDF
   - Please expense receipts after you have spent the money.
   - Please submit one invoice for any related expenses within a calendar month.
   - No rounding.
   - The totals of report should be in USD by default it's in local currency check here how to modify currency for reports in Expensify
   - We require receipts for your expenses.

* Meetup program coming soon

Upstream Development SOW

index : kernel/git/torvalds/linux.git

Linux kernel source tree

master switch

Linus Torvalds

about summary refs log tree commit diff stats

log msg eBPF Fou search

Age	Commit message (Expand)	Author	Files	Lines
6 days	Merge branch 'selftests-bpf-convert-test_tc_edt-sh-into-test_progs'	Alexei Starovoitov	4	-107/+151
6 days	selftests/bpf: do not hardcode target rate in test_tc_edt BPF program	Alexis Lothoré (eBPF Foundation)	2	-3/+4
6 days	selftests/bpf: remove test_tc_edt.sh	Alexis Lothoré (eBPF Foundation)	2	-102/+0
6 days	selftests/bpf: integrate test_tc_edt into test_progs	Alexis Lothoré (eBPF Foundation)	2	-1/+145
6 days	selftests/bpf: rename test_tc_edt.bpf.c section to expose program type	Alexis Lothoré (eBPF Foundation)	2	-2/+3
2025-11-06	selftests/bpf: Use start_server_str rather than start_reuseport_server in tc_...	Alexis Lothoré (eBPF Foundation)	1	-15/+13
2025-11-06	selftests/bpf: Systematically add SO_REUSEADDR in start_server_addr	Alexis Lothoré (eBPF Foundation)	1	-1/+7
2025-10-31	selftests/bpf: Add checks in tc_tunnel when entering net namespaces	Alexis Lothoré (eBPF Foundation)	1	-46/+88
2025-10-31	selftests/bpf: Skip tc_tunnel subtest if its setup fails	Alexis Lothoré (eBPF Foundation)	1	-2/+2
2025-10-31	Merge branch 'selftests-bpf-integrate-test_xsk-c-to-test_progs-framework'	Alexei Starovoitov	6	-2632/+3092
2025-10-31	selftests/bpf: test_xsk: Integrate test_xsk.c to test_progs framework	Bastien Curutchet (eBPF Foundation)	5	-3/+163
2025-10-31	selftests/bpf: test_xsk: Isolate non-CI tests	Bastien Curutchet (eBPF Foundation)	2	-17/+32
2025-10-31	selftests/bpf: test_xsk: Don't exit immediately on allocation failures	Bastien Curutchet (eBPF Foundation)	3	-42/+110
2025-10-31	selftests/bpf: test_xsk: Don't exit immediately if validate_traffic fails	Bastien Curutchet (eBPF Foundation)	1	-4/+8
2025-10-31	selftests/bpf: test_xsk: Don't exit immediately when workers fail	Bastien Curutchet (eBPF Foundation)	1	-34/+76
2025-10-31	selftests/bpf: test_xsk: Don't exit immediately when gettimeofday fails	Bastien Curutchet (eBPF Foundation)	1	-4/+4
2025-10-31	selftests/bpf: test_xsk: Don't exit immediately when xsk_attach fails	Bastien Curutchet (eBPF Foundation)	1	-9/+19
2025-10-31	selftests/bpf: test_xsk: Add return value to init_iface()	Bastien Curutchet (eBPF Foundation)	3	-6/+11
2025-10-31	selftests/bpf: test_xsk: Release resources when swap fails	Bastien Curutchet (eBPF Foundation)	1	-1/+6
2025-10-31	selftests/bpf: test_xsk: Wrap test clean-up in functions	Bastien Curutchet (eBPF Foundation)	1	-12/+24
2025-10-31	selftests/bpf: test_xsk: fix memory leak in testapp_xdp_shared_umem()	Bastien Curutchet (eBPF Foundation)	1	-1/+22
2025-10-31	selftests/bpf: test_xsk: fix memory leak in testapp_stats_rx_dropped()	Bastien Curutchet (eBPF Foundation)	1	-0/+7
2025-10-31	selftests/bpf: test_xsk: Fix __testapp_validate_traffic()'s return value	Bastien Curutchet (eBPF Foundation)	1	-1/+4
2025-10-31	selftests/bpf: test_xsk: Initialize bitmap before use	Bastien Curutchet (eBPF Foundation)	1	-0/+4
2025-10-31	selftests/bpf: test_xsk: Split xskxceiver	Bastien Curutchet (eBPF Foundation)	5	-2634/+2738
2025-10-29	selftests/bpf: Remove test_tc_tunnel.sh	Alexis Lothoré (eBPF Foundation)	2	-321/+0
2025-10-29	selftests/bpf: Integrate test_tc_tunnel.sh tests into test_progs	Alexis Lothoré (eBPF Foundation)	2	-19/+693
2025-10-29	selftests/bpf: Make test_tc_tunnel.bpf.c compatible with big endian platforms	Alexis Lothoré (eBPF Foundation)	1	-35/+22
2025-10-29	selftests/bpf: Add tc helpers	Alexis Lothoré (eBPF Foundation)	3	-93/+74

Upstream Development

index : kernel/git/torvalds/linux.git

Linux kernel source tree

about summary refs log tree commit diff stats

author	Alexis Lothoré (eBPF Foundation) <alexis.lothore@bootlin.com>	2025-11-28 23:27:19 +0100
committer	Alexei Starovoitov <ast@kernel.org>	2025-11-29 09:37:41 -0800
commit	b0f82e7ab6fb2f8501ef87ae928cbf7358d7845e (patch)	
tree	b349f81a8b8e95c9b5241b8e771c14e774222e36	
parent	4b4833acc63e9c8ea9d5897ee84b694f30b23882 (diff)	
download	linux-b0f82e7ab6fb2f8501ef87ae928cbf7358d7845e.tar.gz	

selftests/bpf: integrate test_tc_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>
```

Diffstat

-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

2 files changed, 145 insertions, 1 deletions

```
diff --git a/tools/testing/selftests/bpf/prog_tests/test_tc_edt.c b/tools/testing/selftests/bpf/prog_tests/test_tc_edt.c
new file mode 100644
index 0000000000000..9ba69398eec49c
--- /dev/null
+++ b/tools/testing/selftests/bpf/prog_tests/test_tc_edt.c
@@ -0,0 +1,144 @@
```

bootlin

Home Engineering Training Docs

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 !

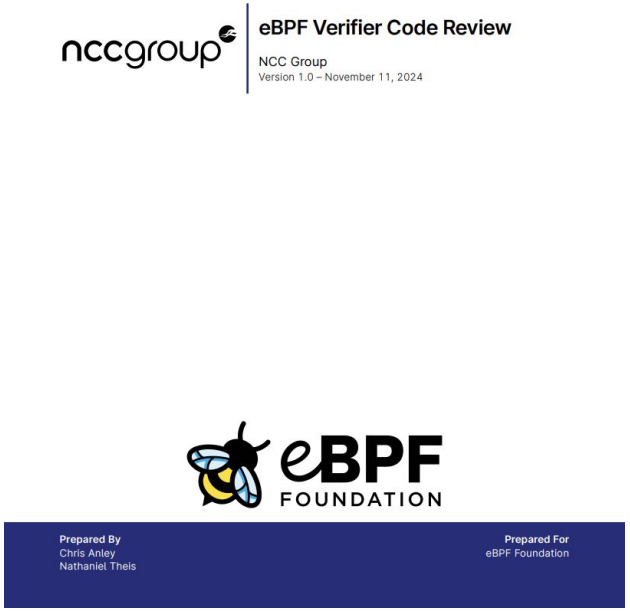
Upstream Development - BTF, Trampolines, and CI

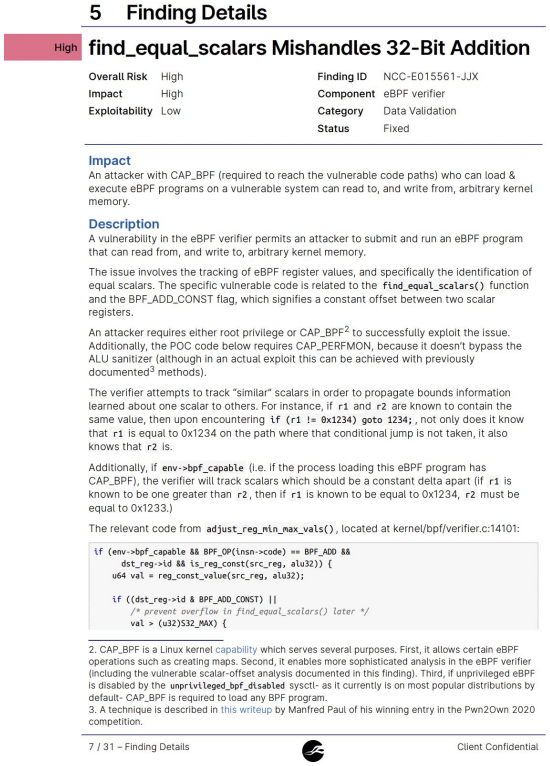
- **Runtime Type Information via BTF**: Solved DWARF size constraints (reducing ~200MB to ~5MB) allowing embedding runtime-loadable type data for dynamic function modeling
- **Low-Overhead Trampolines**: Replaced expensive kprobes with direct function calls acting as a dynamic ABI bridge for fentry/fexit
- **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]





5 Finding Details

High

find_equal_scalars Mishandles 32-Bit Addition

Overall Risk	High	**Finding ID**	NCC-E015561-JJX
Impact	High	**Component**	eBPF 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 `find_equal_scalars()` function and the BPF_ADD_CONST flag, which signifies a constant offset between two scalar registers.

An attacker requires either root privilege or CAP_BPF[2] to successfully exploit the issue. Additionally, the POC code below requires CAP_PERFMON, because it doesn't bypass the ALU sanitizer (although in an actual exploit this can be achieved with previously documented[3] 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 (r1 != 0x1234) 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.

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:

```
if (env->bpf_capable && BPF_OP(insn->code) == BPF_ADD &&
      dst_reg->id && is_reg_const(src_reg, alu32)) {
    u64 val = reg_const_value(src_reg, alu32);

    if ((dst_reg->id & BPF_ADD_CONST) ||
        /* prevent overflow in find_equal_scalars() later */
        val > (u32)S32_MAX) {
```

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` sysctl- as it currently is on most popular distributions by default- CAP_BPF is required to load any BPF program.
3. A technique is described in this writeup by Manfred Paul of his winning entry in the Pwn2Own 2020 competition.

Upstream Security Funding

$225,000 grant from Alpha-Omega for:

- Defensive runtime tooling
  - Enabling sanitizers (like ASan and UBSan) for JIT-compiled eBPF programs to validate memory accesses
- Auditing
  - Security review of x86-64, ARM64, and RISC-V JIT compilers, specifically targeting vulnerabilities in instruction encoding, register allocation, and immediate value handling

How can we improve eBPF security?

- XXX

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

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

Announcements Blog

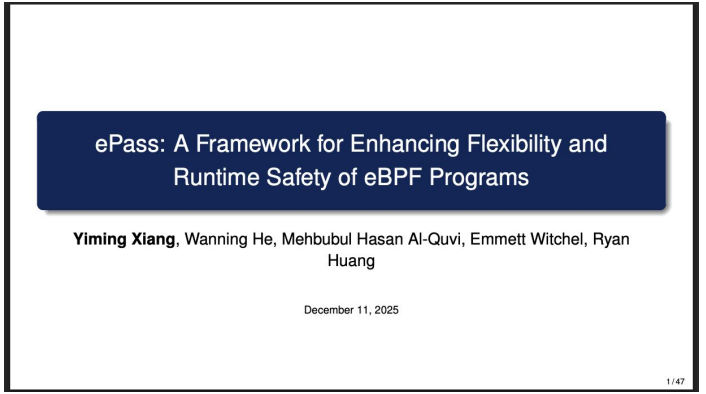
eBPF Foundation Awards $100,000 in Research Grants to Advance eBPF Safety and Efficiency

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

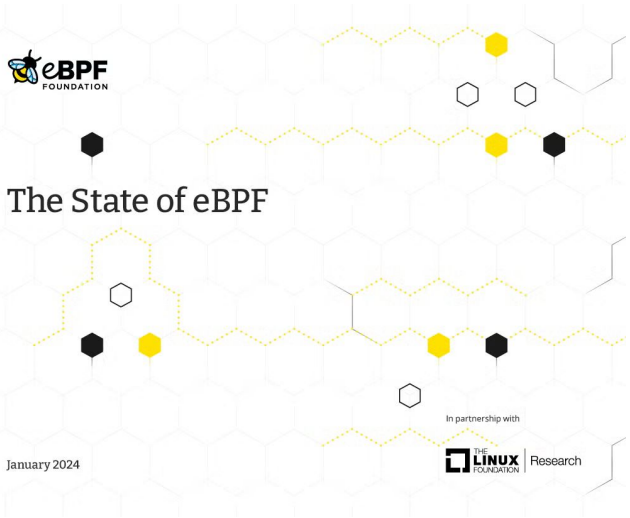
What research is important?

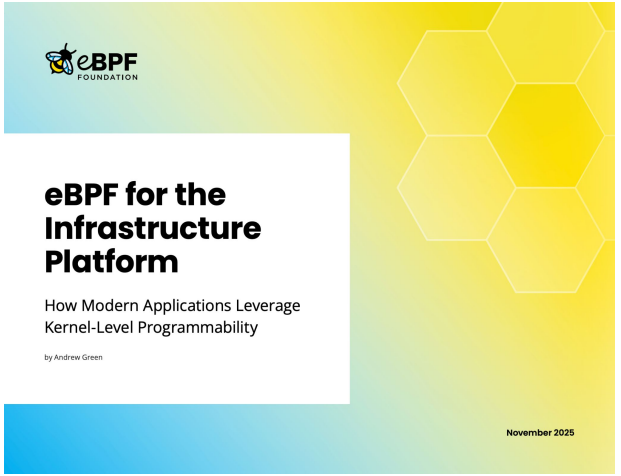
- XXX

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 content platforms around the world at massive scale, faced significant challenges in ensuring performance and stability across its data centers. With over a million servers running containerized applications, the company required a networking solution that could handle high throughput while maintaining stability. By leveraging eBPF technology, Bytedance successfully implemented a decentralized networking solution that improved efficiency, scalability, and performance.

CHALLENGE

As Bytedance scaled its operations, its existing container networking solution, which relied on virtual Ethernet devices, began showing limitations. Key challenges included:

- **Performance Bottlenecks:** The virtual Ethernet-based solution introduced soft-interrupt bottlenecks in the network stack, causing inefficiencies.
- **Stability Concerns:** Ensuring stability at such a large scale was critical, but unproven solutions posed risks in production environments.
- **Kernel Version Constraints:** Upgrading to the required kernel version (6.8) was not immediately feasible due to operational constraints.

To address these challenges, Bytedance required a robust, scalable, 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 container networking stack. Key steps in the implementation included:

- **Introducing netkit:**
  - Bytedance adopted netkit, an eBPF-powered networking device introduced in version 6.8 of the Linux kernel, which provides superior performance over traditional virtual Ethernet in container networking.
  - To accommodate existing constraints and legacy deployments, netkit was backported to kernel version 5.15, ensuring compatibility with Bytedance's infrastructure.
- **Rolling Upgrade Strategy:**
  - A carefully orchestrated upgrade process ensured minimal disruption. Both the container networking interface and kernel upgrades were managed independently, 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.
- **Handling Failures:**
  - Fallback mechanisms were implemented to handle scenarios where netkit or its associated eBPF programs failed, to ensure uninterrupted service.

RESULTS

The deployment of eBPF and netkit delivered significant improvements across Bytedance's data centers:

- **Performance Gains:**
  - Eliminated the last soft-interrupt in the network stack, leading to a 10% improvement in throughput.
  - Resolved issues with high CPU load and packet reordering caused by virtual Ethernet, enhancing overall efficiency.
- **Scalability and Stability:**
  - Successfully deployed netkit across dozens of clusters, demonstrating its reliability and readiness for broader adoption.
- **Operational Benefits:**
  - Simplified the networking stack, reducing maintenance overhead and improving system observability.

FUTURE PLANS

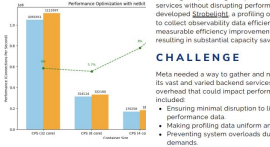
Bytedance plans to further explore th
including:

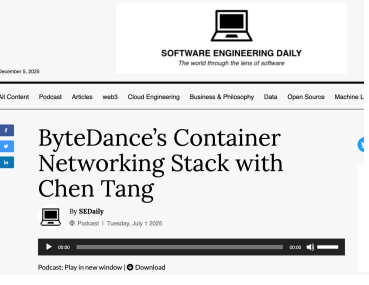
- **Hardware Offloading:** Combining offloading to achieve even greate
- **Broader Use Cases:** Expanding eB container networking to other area

CONCLUSION

Bytedance's adoption of eBPF and ne
transformative potential of this techn
scale networking challenges. By rede
and leveraging the flexibility of eBPF,
measurable performance improveme
and scalability. This deployment serv
capability to drive innovation in mod

To learn more about their use of netk
eBPF Summit.





META'S STROBELIGHT LEVERAGES EBPF TO REDUCE CPU CYCLES AND SERVER DEMANDS BY UP TO 20%

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 Strobelight, a profiling orchestrator that leverages eBPF to collect observability data efficiently. This solution has driven measurable 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 introducing overhead that could impact performance. The key challenges included:

- Ensuring minimal disruption to live services while collecting performance data.
- Making profiling data uniform and easily interpretable.
- Preventing system overloads due to excessive data storage demands.
- Supporting multiple kernel versions across Meta's infrastructure.

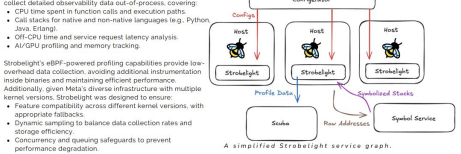
SOLUTION

To overcome these challenges, Meta implemented Strobelight, a sophisticated profiling orchestrator that integrates multiple profiling tools, including eBPF. Strobelight enables engineers to collect detailed observability data out-of-process, covering:

- CPU time spent in function calls and execution paths.
- Call stacks for native and non-native languages (e.g., Python, Java, Erlang).
- Off-CPU time and service request latency analysis.
- AI/GPU profiling and memory tracking.

Strobelight's eBPF-powered profiling capabilities provide low-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 storage efficiency.
- Concurrency and queuing safeguards to prevent performance degradation.



A simplified Strobelight service graph.

ANT GROUP SECURES THEIR PLATFORM WITH KATA CONTAINERS AND eBPF FOR FINE GRAINED CONTROL

OVERVIEW

Ant Group needed to strengthen security for its large scale containerized workloads, which power their critical online services and AI applications. Traditional container workload protection solutions left gaps around container escape, inconsistent kernel support, and policy management complexity. To address these challenges, Ant Group developed the Cloud Workload Protection Platform (AntCWPP), combining Kata containers with eBPF. This solution provides strong workload isolation, fine-grained security auditing, and real-time behavioral enforcement. AntCWPP has successfully eliminated the risk of container escape, reduced the blast radius of attacks, and delivered flexible policy enforcement in on environments.

LLENGES

al technology leader, Ant Group operates diverse
ized workloads across large-scale production clusters.
orkloads faced a growing set of security challenges:
iner escape and blast radius risks: In traditional
ves (e.g., runc), containers share the host kernel. This
ecture makes it difficult to fully prevent attackers from
ng isolation, and a single compromise can affect
ple workloads.
se security needs: Internet-facing services, internal
ations, and AI agents require different security policies.
sistent kernel versions: Large-scale production
onments contain many kernel versions, making it
tex to deploy and maintain consistent security policies
s all workloads.

ai CWPP solutions built on kernel modules or host-level
oduced operational challenges in these environments,
ring the need for an approach that could combine
olation with flexible, fine-grained enforcement.

META

RESULTS

The deployment of eBPF within Strobelight has led to significant efficiency gains, including:

- 15,000 servers' worth of annual capacity savings from a 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 production.
- Dynamic sampling mechanisms, optimizing profiling rates without overloading storage systems.

WHY EBPF?

eBPF was selected as the core technology due to its low overhead, which ensured minimal impact on targeted processes; flexibility, with numerous attach points and built-in kernel helpers; and lack of requirements for additional instrumentation, simplifying deployment.
Additionally, unlike traditional profiling tools that require added instrumentation 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 eBPF to further enhance observability, particularly in:

- AI/ML workloads.
- Advanced memory tracking.
- More complex efficiency analyses for improved resource utilization.
- Open sourcing Strobelight's profilers and libraries for broader adoption within the open source community.

RAKUTEN MOBILE ADOPTS EBPF TO STRENGTHEN ANOMALY DETECTION AND SECURITY IN CLOUD-NATIVE TELECOM NETWORKS

OVERVIEW

As mobile networks transition to virtualized and cloud-native architectures, traditional monitoring tools struggle to keep pace. To address this challenge, Rakuten Mobile adopted open source eBPF technology, specifically 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 Rakuten's 5G infrastructure, enhancing both performance and security.

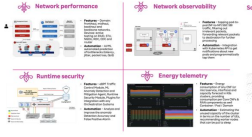
CHALLENGES

- Overwhelming Alert Volume: Static rules and thresholds triggered large volumes of false positives, exhausting operational resources and reducing system trust.
- Poor Adaptability: Legacy solutions lack the flexibility to adapt to evolving network topologies, routing paths, and zero-day attacks.
- Limited Real-Time Insight: Traditional passive monitoring tools couldn't provide granular visibility or timely feedback, especially across distributed, containerized environments.
- Performance Impact: Instrumenting the network without adding significant overhead remained a constant tradeoff.

SOLUTION

Rakuten Mobile developed Sauron eBPF, a lightweight, flexible framework integrated with AI/ML to address the visibility and security gaps in telecom networks. It provides:

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



- Observability (Traffic Mirr
Native Network Function
filtering. Fine-grained CN
optimization. Reduced ov
time insights.
- Security: Linux-based tra
policy-driven alerts on th
malicious activity.
- Enhanced AI/ML Capabil
time threat detection from
learning and adaptation b
SOAR platforms.

RESULTS

- Hop-by-Hop Visibility: I
performance enabled p
- Reduced False Positive
allowing teams to focus
automated monitoring.
- Improved Threat Detec
respond to previously u
including zero-day thre
- Increased Efficiency: D
and intervention reduct

FUTURE PLA

- Broader eBPF Use Cas
control plane behavior.
- Advanced Threat Intelli
technologies. Joint Embedding Predictive Architectures (JEPAs), and reinforcement learning to enhance autonomous root cause analysis.
- Cross-Domain Correlation: Linking telemetry from multiple domains (compute, storage, network) for full-stack observability.

CONCLUSION

Rakuten Mobile's adaptation of eBPF for telecom use demonstrates the transformative power of combining real-time kernel-level telemetry with AI-driven analytics. By bridging the gap between static monitoring systems and dynamic, distributed mobile networks, Rakuten is not only improving performance and security, but also laying the foundation for intelligent, self-healing 5G and 6G networks.

To learn more about their use of Sauron eBPF, check out the article on IEEE Access.

ALIBABA CLOUD LEVERAGES eBPF FOR ADAPTIVE LAYER 7 LOAD BALANCING

OVERVIEW

Alibaba Cloud is one of the largest public cloud providers, operating a globally distributed infrastructure for millions of tenants. As their Layer 7 (L7) load balancers scaled to process over 10 million requests per second across diverse workloads, traditional Linux I/O event notification mechanisms struggled to keep up. To solve this problem, Alibaba Cloud developed Hermes, an intelligent, userspace-directed I/O event notification framework powered by eBPF. By integrating real-time feedback from application workers directly into kernel scheduling decisions, Hermes enables adaptive connection management, fine-grained performance control, and greater infrastructure efficiency. This eBPF-driven approach strengthens the stability and scalability of Alibaba Cloud's global network infrastructure, reducing operational overhead, lowering cloud infrastructure costs, and enhancing the tenant experience.

CHALLENGE

Alibaba Cloud's L7 load balancers must sustain high throughput 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 connection concentration on recently active workers, while SO_REUSEPORT's static hashing failed to detect overloaded or hung processes.
- **Operational Risk**: Modifying the kernel was not feasible at scale, where even minor bugs could cause large-scale service disruptions.
- **Performance Sensitivity**: Maintaining low latency under multi-tenant workloads required extremely low-overhead scheduling and real-time feedback.
- **Tenant Performance Isolation**: Load-balancing mechanisms failed to evenly distribute resources across tenants, causing performance imbalances in shared environments.

To address these issues, Alibaba Cloud sought a solution that could safely extend kernel scheduling behavior while incorporating dynamic, userspace-driven intelligence.

SOLUTION

Alibaba Cloud created Hermes, a userspace-directed I/O event notification framework that leverages eBPF to augment the Linux kernel with real-time worker metrics. This framework enables closer, more coordinated collaboration between userspace and the kernel, ensuring that connection dispatch decisions reflect actual workload conditions:

- **Closed-Loop Scheduling**: Userspace workers continuously publish metrics—such as availability, pending events, and active connections to shared memory. An eBPF program reads these metrics and dynamically selects the most suitable worker.
- **Lock-Free Synchronization**: Worker status is exchanged through atomic operations on eBPF array maps, enabling high-speed updates without locks or contention.

RESULTS

Hermes has delivered measurable and sustained improvements across Alibaba Cloud's L7 load balancing systems:

- **19% reduction in unit infrastructure cost** by increasing safe CPU utilization thresholds from 30% to 40% without compromising stability.
- **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 handling **10 million requests per second** for more than two years.
- **Enhanced reliability** and reduced operational overhead.

WHY eBPF?

- **Scalable Metric Design**: Lightweight metrics allow real-time updates per I/O event with negligible overhead.
- **Two-Stage Load Distribution**: Userspace preselects eligible workers, and the kernel refines selection using a hash of the connection context.
- **Production-Safe Implementation**: The kernel dispatch logic running in eBPF prevents loops, recursion, and unbounded execution, using bitwise operations for minimal CPU overhead, and is production-ready at scale.

This architecture enables Hermes to adapt continuously to runtime conditions while preserving system stability and efficiency.

DATADOG USES EBPF TO IMPROVE NETWORK OBSERVABILITY ACCURACY AND PERFORMANCE

OVERVIEW

Datadog, Cloud Network Monitoring (CNM) collects Layer 4 network data, which is then combined with other data streams during processing. This allows users to not only dive into their raw network telemetry but also to slice and dice the data using unconventional criteria, like container and service names. CNM provides this flexibility by being able to attribute network connections to containers and correctly identify IP addresses, specifically after address translation. Earlier versions of the Datadog Agent performed these actions by relying on Netlink and container runtime APIs with periodic polling, which caused performance degradation and data loss. Using eBPF, allowed Datadog to overcome these issues and obtain both container and IP data more efficiently.

CHALLENGE

- **Network Address Translation**: Originally, CNM acquired Network Address Translation (NAT) data by listening to connection tracking (conntrack) via Netlink. This process was performance-intensive, especially on hosts with high connection churn – so much so that Datadog had to engineer a throttling mechanism for these updates. This resulted in dropping data when throttling limits were reached, which in turn hampered the product's ability to accurately attribute host and container data to IP addresses. Since the translated IP addresses can be reused, so too can connections in the data pipeline, leading to inaccuracies.
- **Process to Container Attribution**: The Agent was already using eBPF to attribute process IDs (PIDs) to network connections. However, attributing PIDs to containers was done in user space by periodically polling the container runtime APIs on the host (e.g., Docker). This meant that processes that were short-lived (i.e., shorter than the polling interval) would not always show up in the polled data from the container runtimes, so no container attribution was possible for some connections.
- **Missing Process Launches**: The Datadog Agent was also using Netlink updates for process launches to hook uprobes. Similar to conntrack updates, this process was performance-intensive and added a considerable operational burden to the product.

To address these challenges, Datadog needed a performant solution with a focus on minimal data loss and maximum accuracy.

SOLUTION

Datadog used eBPF to come up with a couple of approaches that helped the company mitigate the problems noted above.

eBPF-based Connection Tracker

Using eBPF kprobes, the team at Datadog hooked into connection tracking table insertions, storing them in their own eBPF map for later lookups from user space.

Process Event Data Stream

This approach uses eBPF to get updates about processes, including attributing container IDs to a process ID. This process data—which includes the container ID, if applicable—is then sent to user space where it is cached for easy querying later.

RESULTS

eBPF-based Connection Tracker

Using kprobes to insert updates into Datadog's map ensured that they were only limited by the size of the eBPF map, in addition to completely forgoing the performance penalty from the use of Netlink. CPU usage dropped by roughly 35% with use of the eBPF-based connection tracker. This helped them eliminate throttling updates and avoid dropping data, resulting in increased attribution of IP addresses to other, more valuable customer constructs, such as service names.

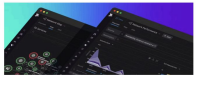
Process Event Data Stream

Using eBPF allowed Datadog to get real-time updates about the process life cycle, picking up even short-lived processes that were missed by the polling approach. This meant more reliable attribution of processes to containers, leading to a richer query experience for customers in the product. Although this real-time approach increased CPU usage, Datadog's team determined the tradeoff was acceptable because of the accuracy gains.

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

NEXT STEPS

Datadog plans on using eBPF to correlate more connection data closer to when they first see it, instead of in user space. In addition, the company continues to invest in using eBPF to collect more network data and metrics as they enable new features in their network monitoring products.



Community Building Fellowships

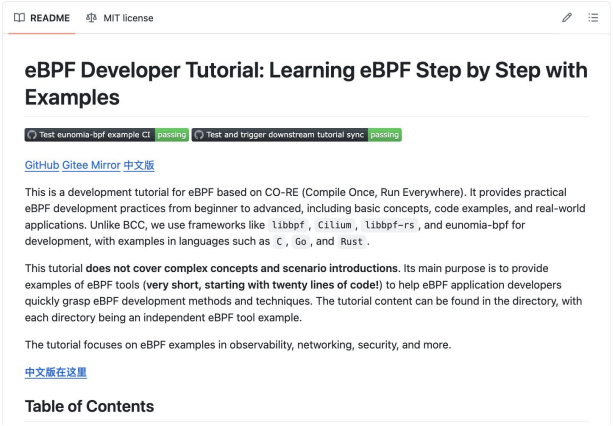
Uncategorized

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).


README MIT license

eBPF Developer Tutorial: Learning eBPF Step by Step with Examples

Test eunomia-bpf example CI passing Test and trigger downstream tutorial sync passing

GitHub Gitee Mirror 中文版

This is a development tutorial for eBPF based on CO-RE (Compile Once, Run Everywhere). It provides practical eBPF development practices from beginner to advanced, including basic concepts, code examples, and real-world applications. Unlike BCC, we use frameworks like libbpf, Cilium, libbpf-rs, and eunomia-bpf for development, with examples in languages such as C, Go, and Rust.

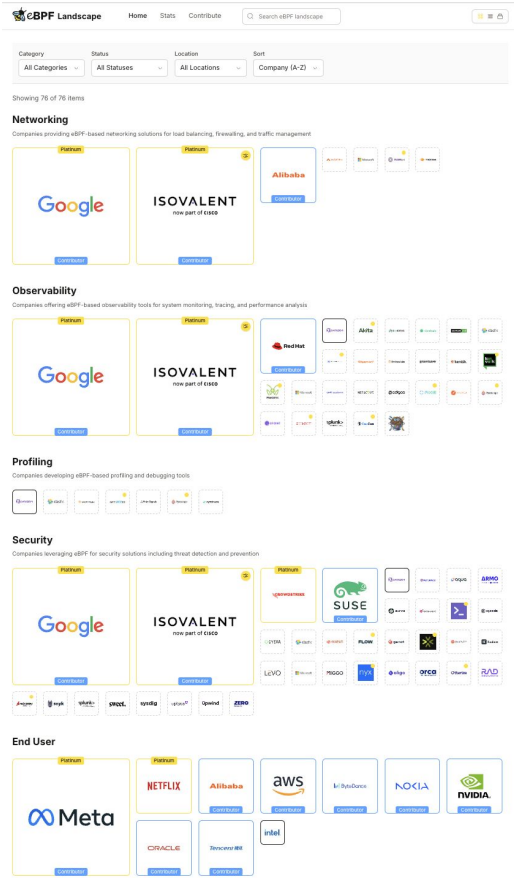
This tutorial **does not cover complex concepts and scenario introductions**. Its main purpose is to provide examples of eBPF tools (**very short, starting with twenty lines of code!**) to help eBPF application developers quickly grasp eBPF development methods and techniques. The tutorial content can be found in the directory, with each directory being an independent eBPF tool example.

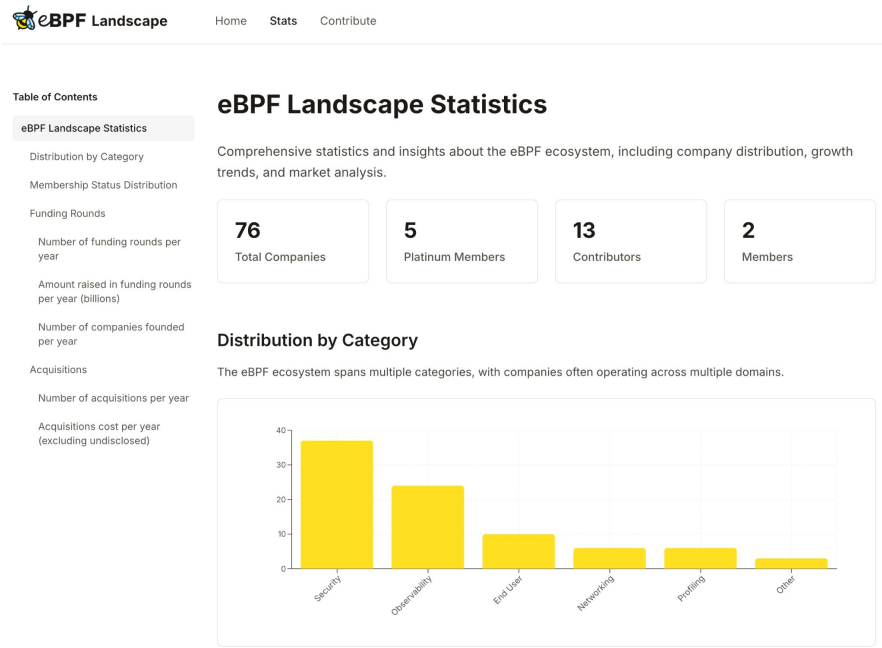
The tutorial focuses on eBPF examples in observability, networking, security, and more.

中文版在这里

Table of Contents

Company Landscape





What information would be helpful to surface?

- XXX

Join the Foundation

eBPF FOUNDATION

Who are our members

Platinum

Silver

Join the Foundation

How you benefit from membership

Strategic Influence

Take part in shaping the technical roadmap and key decisions for the future of eBPF.

Cross-Platform Collaboration

Work openly with leading industry leaders and developers in a neutral, collaborative environment.

Security & Research

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

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 visibility.

- 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*

Get started

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

Discussion