CYBERAID

AI-Driven Cybersecurity for Financial Service Providers

Module 1
eBPF Fundamentals and the Visibility Gap

CyberAId Learning Centre – eBPF Tutorial Course

October 2026

Co-funded by the European Union

ECCC EUROPEAN CYBERSECURITY COMPETENCE CENTRE

The project funded under Grant Agreement No 101249596 is supported by the European Cybersecurity Competence Centre

CYBERAID

What is eBPF?

Safe

A verifier checks every program before loading: bounded execution, valid memory access, no kernel crashes

Fast

JIT-compiled to native code; events are filtered in the kernel so only relevant data reaches user space

Dynamic

Programs load and unload at runtime: no reboot, no kernel rebuild, no application restart

Programmable

Maps and ring buffers share state and events between kernel programs and user-space collectors

eBPF does for the kernel what JavaScript did for the browser: safe, event-driven programmability at runtime.

CYBERAID

From classic BPF to today

1

1992

Classic BPF: in-kernel packet filtering, e.g. tcpdump

2

2014

Extended BPF (eBPF) merged into the Linux kernel

3

2015–2019

Tooling matures: BCC, bpftrace, Cilium, libbpf

4

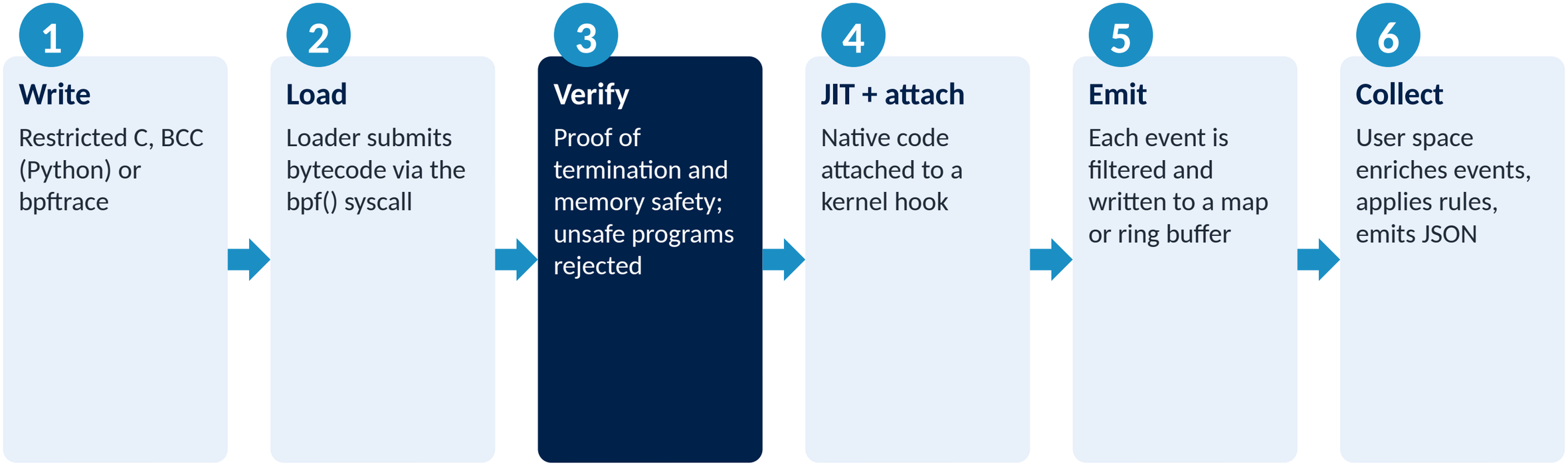
Today

Mainstream for cloud-native networking, observability and runtime security

Simplified timeline; see ebpf.io for the project landscape.

CYBERAID

How an eBPF program runs



The monitored application is never modified or restarted.

CYBERAID

Hook points: where eBPF can observe

Hook	What it observes	Financial example
kprobes	Kernel function entry and return; flexible, tied to kernel internals	tcp_v4_connect: every outbound connection from a payment API
Tracepoints	Stable, documented kernel events; preferred when available	sched_process_exec: a shell spawned by the banking app
uprobes	Functions in user-space binaries and libraries	TLS library calls: visibility before encryption, strict controls
XDP / TC	Packet path at the driver or traffic-control layer	Drop scanning or flood traffic before the stack
BPF-LSM	Security hooks: observe and enforce	Block unexpected reads of key material or DB files
cgroup / socket	Containers, pods or sockets	Identity-aware, per-pod network policy

The visibility gap in financial systems

Application logs

Show what the application chooses to say; a compromised service can stay silent

Network IDS and packet capture

See traffic, not the process behind it; encryption limits visibility into content

auditd

Strong process context, but can be heavy under high event volumes

eBPF

Host-level view of processes, connections and files, with low overhead and no application changes

Indicative grouping; eBPF complements, and does not replace, existing SIEM, log and network sources.

CYBERAID

eBPF compared with traditional monitoring

Approach	Network visibility	Low overhead	Process context	Host stability
Application logs	Weak	Strong	Weak	Strong
auditd	Partial	Partial	Strong	Strong
Packet capture / network IDS	Strong	Partial	Weak	Strong
Kernel modules / legacy agents	Strong	Partial	Strong	Weak
eBPF	Strong	Strong	Strong	Strong

Qualitative, indicative comparison for typical deployments; not a benchmark.

Why it matters for financial services

1. Minimal interference with payment, banking, authentication and trading services
2. Predictable communication patterns make allow-lists effective
3. Host-level view of connections and processes, even for encrypted traffic
4. Metadata, not payloads: supports data minimisation and GDPR
5. Evidence for DORA and NIS2 monitoring and incident reporting

Key takeaways

- eBPF runs verified, sandboxed programs inside the Linux kernel
- Safe, fast, dynamic and programmable: no kernel modules, no application changes
- Hooks define what can be observed; tracepoints are preferred when available
- It closes the visibility gap left by logs, network sensors and auditd
- Particularly relevant for financial systems: low interference and metadata-only collection



CYBERAID

Thank you!

Connect With CyberAID:
LinkedIn: CyberAID

Next: Module 2 – State of the Art, Tooling and Positioning

CyberAID has received funding from the DIGITAL EUROPE Programme under grant agreement No. 101249596