CYBERAID

AI-Driven Cybersecurity for Financial Service Providers

Module 2
State of the Art, Tooling and Positioning

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

The eBPF tooling stack

Tool	What it is	Best for	Role in CyberAId
BCC	Python/C front end for custom tracers	Rapid prototyping	PoC: TCP tracer, process monitor
bpftrace	High-level one-liner language	Ad-hoc investigation	Analyst toolbox in incidents
libbpf + CO-RE	C / Rust / Go libraries	Portable production agents	Path to hardened agents
Cilium + Hubble	Kubernetes CNI and flow observability	Identity-aware policy	Planned for Kubernetes pilots
Tetragon / Falco	Runtime security engines with policies	Process, file and network threats	Reference designs for rule libraries

CYBERAID

From prototype to production

1

Prototype

BCC tracers in a simulated financial stack

2

Integrate

Wazuh decoders, rules and OpenSearch dashboards

3

Scale

CO-RE agents and Cilium for Kubernetes pilots

Maturity path; the third stage depends on the outcome of the pilot benchmarks.

CYBERAID

The eBPF security landscape

1 Open-source runtime security

Falco, Tetragon, Tracee: policy-driven detection on syscalls, processes, files and network

2 Commercial EDR/XDR sensors

Several Linux sensors use eBPF for process and network telemetry, e.g. CrowdStrike, Defender for Endpoint, Elastic Defend

3 Cloud-native networking and observability

Cilium/Hubble, Pixie, Parca: flows, identities and profiling in Kubernetes

4 Research and emerging work

ML on kernel telemetry, in-kernel inference, LLM-assisted eBPF program generation

Indicative grouping; product names are examples of each category, not an endorsement or ranking.

CYBERAID

Platform evolution

CO-RE and BTF

Compile once, run everywhere: portability across kernel versions

BPF-LSM

Enforcement hooks on Linux Security Module points, kernel 5.7 and later

Ring buffer

Efficient event delivery from kernel to user space, kernel 5.8 and later

Fine-grained privileges

CAP_BPF and CAP_PERFMON instead of full root

Hardening

Restricted unprivileged eBPF and tighter control over program loading in recent kernels

Beyond Linux

Open-source eBPF for Windows with a similar programming model

Kernel versions are indicative; verify against the distributions used in each pilot.

CYBERAID

Typical strengths and limitations

Approach	Typical strengths	Typical limitations
Logs and SIEM rules	Cheap, familiar, flexible	Depend on what applications record
Network IDS / capture	Strong traffic visibility	No process context; encryption limits content
Host agents / kernel modules	Deep host visibility	Stability risk, heavy maintenance
eBPF-based sensors	Process + network context, low overhead, verified programs	Kernel support and privileges needed; data volume needs filtering

General tendencies, not statements about any specific product.

CYBERAID

Risks: the security of eBPF itself

1. Verifier vulnerabilities have enabled privilege escalation on some kernels: keep kernels patched
2. eBPF can be abused for stealth, e.g. BPFDoor and Symbiote: restrict and monitor program loading
3. Loading programs needs privilege: use CAP_BPF and CAP_PERFMON, change control
4. kprobes depend on kernel internals: prefer tracepoints and CO-RE
5. High event rates can overload collectors: filter in the kernel

How the CyberAId monitor compares

Dimension	Open-source engines	Commercial EDR/XDR	CyberAId monitor
Focus	Generic runtime / Kubernetes	Broad endpoint protection	Financial profiles: payment, banking, auth, trading
Openness	Open source	Proprietary	Open-source stack
SIEM integration	Connectors	Vendor console	Wazuh decoders + financial rules
Agent integration	Not designed for it	Vendor-specific	Governed source and tool for agents
Privacy	Configurable	Varies	Metadata only by design

Indicative and qualitative; not a benchmark. The CyberAId monitor is a proof of concept.

Limitations we state openly

1. Proof of concept validated on a simulated financial stack, not on production systems
2. Overhead to be benchmarked under peak load in the pilots
3. Thresholds are examples and need calibration for each institution
4. Kernel version and BTF support are prerequisites for portable agents
5. Linux-centric and metadata-only: other threats need other components

Key takeaways

- The eBPF ecosystem is a stack: BCC and bpftrace, libbpf with CO-RE, Cilium, Tetragon, Falco
- Open-source engines, commercial sensors and research all use eBPF
- The platform is maturing: portability, enforcement hooks, hardening
- eBPF has its own risks and needs governance
- CyberAId's differentiators: financial profiles, openness, integration, privacy



Thank you!

Connect With CyberAID:
LinkedIn: CyberAID

Next: Module 3 – The CyberAId Kernel-Level Monitor in Practice

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