



CYBERAID

Cyber Defense Simulator: eBPF Operations

Interactive Knowledge Validation
for Financial Infrastructure

SYSTEM PROTOCOL: INITIATED // TARGET: eBPF TELEMETRY //
TELEMETRY // CLEARANCE: CYBERAID ARCHITECT

MODULE MANIFEST

Phase 1 // Initialization

Phase 2 // Instrumentation

Phase 3 // Threat Interception

Phase 4 // Operational Readiness

Mission Briefing: Validating Kernel Visibility

You have been assigned to monitor a simulated financial environment—including payment, core banking, and authentication services. To detect advanced threats without altering the underlying applications, you will deploy eBPF.

The following diagnostic will test your operational command of eBPF. Each section presents an incoming scenario. Pause, formulate your strategy, and proceed to the telemetry debrief to verify your approach.

STATUS: READY // AWAITING OPERATOR INPUT

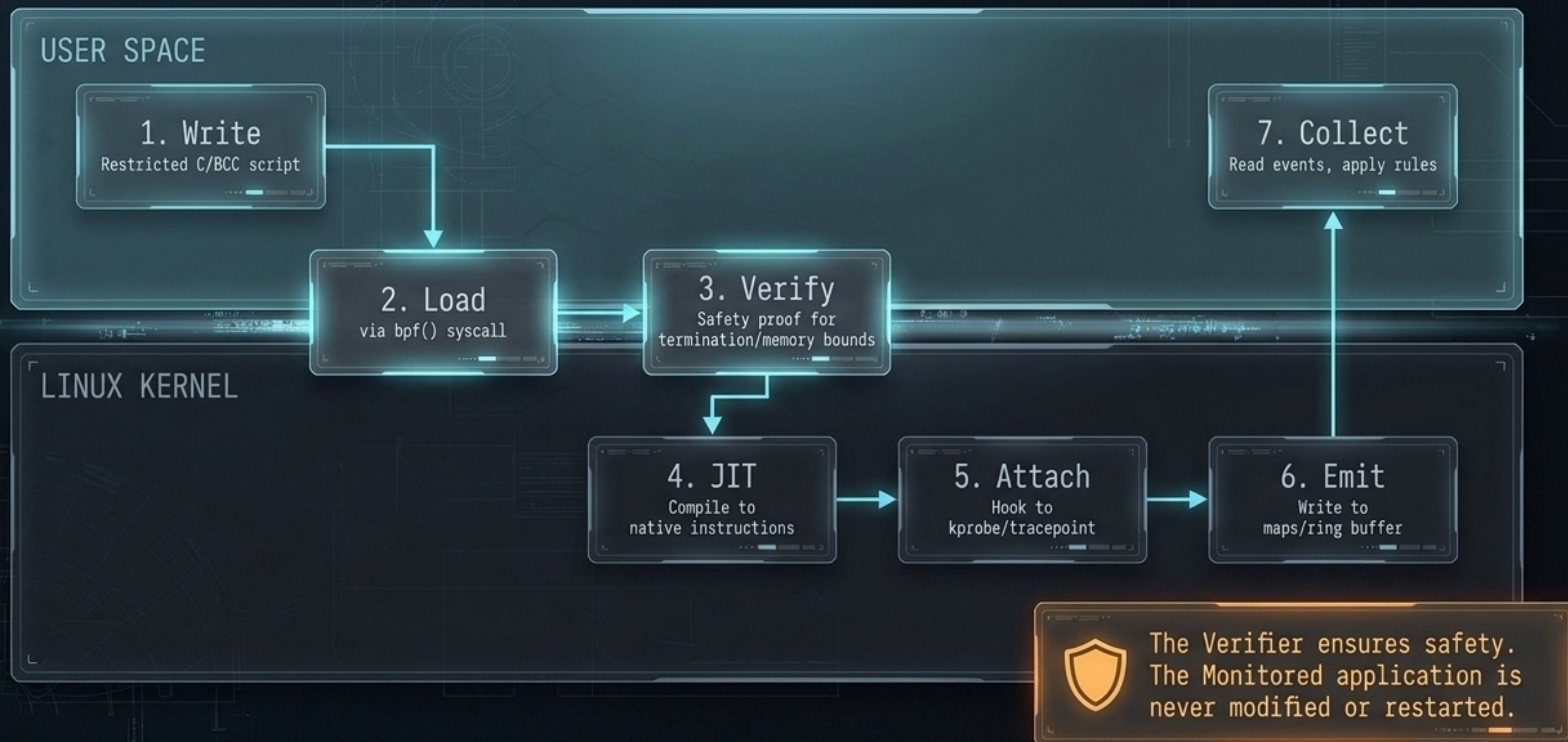
⚠ System Prompt // Pipeline Assembly

An analyst is attempting to deploy a custom BCC Python tracer to monitor outbound connections from a payment API, but the script is failing to execute in the kernel.

The deployment sequence is compromised. Mentally reorder these seven execution steps into the correct kernel/user-space pipeline before proceeding:



Telemetry Decoded // The Execution Boundary



System Prompt // The Visibility Gap

Application Logs



NOMINAL

auditd



NOMINAL



Data Exfiltration Successful

Your financial infrastructure relies exclusively on Application Logs and legacy syscall auditing (auditd). A threat actor successfully exfiltrates data from a core banking host using a reverse shell, bypassing your SIEM entirely.

Look at your current stack. Which two critical visibility dimensions are you missing that allowed this attack to remain undetected?

Telemetry Decoded // The eBPF Advantage Matrix

	Network Visibility	Low Overhead	Process Context	Host Stability
App Logs	 Weak	 Strong	 Weak	 Strong
auditd	 Partial	 Partial	 Strong	 Strong
Packet Capture	 Strong	 Partial	 Weak	 Strong
Kernel Modules	 Strong	 Partial	 Strong	 Weak
eBPF	 Strong	 Strong	 Strong	 Strong

Rule of Thumb: Logs tell you what the application says. eBPF tells you what the system actually did.

Module Loader // Selecting the Toolchain



Rapid Prototyping
(The CyberAID PoC)

BCC

(Python/Lua front-end)

Best for custom TCP tracers
and process monitors.



Incident Triage & Ad-Hoc

bpftrace

(One-liner language)

Best for analyst toolboxes
during active investigations.



Production Scale &
Kubernetes

libbpf + CO-RE
/ Cilium

(with Tetragon for
runtime security)

Best for portable, hardened
agents and identity-aware
network policies.

System Prompt // Threat Vector Interception

A coordinated attack is hitting the trading systems. You need to deploy specific sensors to intercept the telemetry. **Map the correct eBPF hook point to the threat scenario:**

! Threat Profile

T1110: Brute force bursts on authentication ports.

! Threat Profile

T1552: Unauthorized reading of DB key materials.

! Threat Profile

T1041: TLS-encrypted data exfiltration via user-space binaries.

📦 Sensor Config

uprobes (User-space functions)

📦 Sensor Config

kprobes
(Kernel function entry/return)

📦 Sensor Config

BPF-LSM
(Linux Security Module hooks)

Telemetry Decoded // Hook Point Architecture

! Threat Profile

T1110: Brute force bursts on authentication ports.

tcp_v4_connect ->
Captures every outbound connection attempt natively.

≡ Sensor Config

kprobes
(Kernel function entry/return)

! Threat Profile

T1552: Unauthorized reading of DB key materials.

Enforces blocks on unexpected file-opens based on monitoring profiles.

≡ Sensor Config

BPF-LSM
(Linux Security Module hooks)

! Threat Profile

T1041: TLS-encrypted data exfiltration via user-space binaries.

Hooks TLS library calls for visibility before encryption occurs.

≡ Sensor Config

uprobes
(User-space functions)

For high-volume flood traffic, intercept even lower at the **network driver** using **XDP/TC** to drop packets before they hit the stack.

The CyberAID Data Flow // From Kernel to Dashboard

Simulated Financial Environment

Docker, core banking, attack generators

Kernel Hooks

BCC/eBPF capturing TCP connections and file access

Structured Output

Filtering massive kernel noise into structured JSON (PID, IP, timestamp)

Wazuh Integration

Custom decoders applying financial-domain detection rules

OpenSearch Dashboard

Graded alerts: INFO -> CRITICAL

Severity ring



● INFO ● WARNING ● CRITICAL

Real time timeline



Critical alerts

Severity	Name	PID	10:15:30
CRITICAL	Unauthorized DB Access	PID 1234	10:15:30
CRITICAL	Unauthorized DB Access	PID 1234	10:15:30
CRITICAL	Unauthorized DB Access	PID 1234	10:15:30
CRITICAL	Unauthorized DB Access	PID 1234	10:15:30

System Prompt // Production Go/No-Go

The engineering team submits the following operational plan for deploying eBPF into the European payment processor cluster.

PRE-FLIGHT DEPLOYMENT PLAN

- [] We will run the agent with full root privileges.
- [] We will capture and log full transaction payloads for forensic analysis.
- [] We will rely solely on host probes, ignoring Kubernetes pod identities.

Identify **the three critical regulatory and architectural violations** in this deployment plan.

Telemetry Decoded // Deployment Protocol

- [✓] We will run the agent with full root privileges.
Least Privilege. Do not use full root. Use CAP_BPF / CAP_PERFMON where supported.
- [✓] We will capture and log full transaction payloads for forensic analysis.
Privacy by Design (GDPR). Collect metadata only (connections, processes). Never capture or store transaction payloads.
- [✓] We will rely solely on host probes, ignoring Kubernetes pod identities.
Kubernetes Context. Complement host probes with Cilium for identity-aware, per-pod network policies.

Added Requirement: Kernel Readiness & Governance.

Ensure BTF support for portable (CO-RE) programs and retain events for DORA reporting.



CYBERAID

Mission Synthesis // The CyberAID eBPF Philosophy



Safe Visibility

Low-overhead kernel telemetry without touching or modifying the monitored financial systems.

Deep Context

Fusing process execution, network flows, and identity into a single cohesive record.

Actionable Intelligence

Translating raw kernel metadata into structured, DORA-compliant SIEM alerts for rapid forensic correlation.

Threats bypass the application layer. By monitoring the kernel, CyberAID ensures the infrastructure itself becomes the ultimate line of defense.