IronClad Kernel Guardian

Technical White Paper v2.0

Date: November 2025
Author: IronClad Development Team
Architecture: eBPF / Golang / Linux Security Modules (LSM)

1. Executive Summary

In the modern threat landscape, the time gap between a vulnerability disclosure (CVE) and the application of a patch is the "Danger Zone." Traditional Endpoint Detection and Response (EDR) tools often rely on signature matching user-space behavior, which can be bypassed or disabled by rootkits.
IronClad Kernel Guardian is a next-generation **Active Defense System** that operates natively within the Linux kernel. By leveraging **eBPF (Extended Berkeley Packet Filter)** and **LSM (Linux Security Modules)**, IronClad provides a programmable, unbypassable security layer that can quarantine vulnerable processes in real-time, effectively shielding unpatched systems from exploitation until a proper fix is applied.

2. The Problem: The User-Space Gap

Traditional security tools face inherent limitations:

1. **Performance Overhead**: Constant context switching between kernel and user space to analyze system calls degrades server performance.
2. **Race Conditions**: By the time a user-space agent detects a malicious execve() call, the process may have already executed.
3. **Tamper Susceptibility**: Attackers with root privileges can often kill user-space security agents (blindfolding the defender).

IronClad addresses these issues by moving the enforcement logic **into the kernel itself**.

3. Solution Architecture

IronClad employs a split-brain architecture designed for stability and speed.

3.1 The Enforcer (Kernel Space - eBPF)

The core of IronClad is a compiled C program (vuln_detector.bpf.c) loaded into the kernel.

- **Technology**: eBPF (safe, sandboxed virtual machine within the Linux kernel).
- **Hook Point**: lsm/bprm_check_security. This specific LSM hook triggers *before* a process is allowed to execute a new binary.
- **Mechanism**: It checks a high-performance Hash Map (BPF_MAP_TYPE_HASH) to see if the calling PID is marked as "Quarantined."
- **Action**: If a match is found, IronClad returns -EPERM, causing the kernel to deny execution immediately. The malicious payload never runs.

3.2 The Intelligence Engine (User Space - Go)

The control plane is written in Go, providing logic, correlation, and user interaction.

- **Scanner**: Uses gopsutil and direct system calls to map running processes and installed packages (dpkg/rpm) to known vulnerabilities.
- **ExploitDB Integration**: Parses the local exploitdb CSV database to correlate installed software versions with known exploit IDs.
- **Telemetry Aggregator**: Reads from an eBPF Ring Buffer (BPF_MAP_TYPE_RINGBUF) to receive high-performance event streams from the kernel without polling.

3.3 The Dashboard (TUI)

A terminal-based UI (using the Bubble Tea framework) provides real-time situational awareness, allowing operators to visualize threats, system resources, and mitigation status without leaving the terminal.

4. Key Capabilities

Feature	Description	Technical Implementation
Zero-Downtime Mitigation	Apply virtual patches to block exploitation without restarting services.	eBPF Map Updates (Atomic)
Process Quarantine	Prevent specific PIDs from spawning shells (RCE protection).	LSM Hook on execve
Real-Time Telemetry	Nanosecond-precision logging of blocked attempts.	BPF Ring Buffer
Auto-Updating Logic	Automatically pulls new vulnerability signatures.	Git Integration with ExploitDB
Low Overhead	Minimal CPU impact (<1%).	JIT-compiled BPF bytecode

5. Operational Workflow

1. **Initialization**: IronClad starts as a systemd daemon (ironclad.service). It compiles the eBPF agent JIT (Just-In-Time) tailored to the specific kernel headers of the host machine.
2. **Scan Phase**: The Go engine scans the process tree. It identifies, for example, vsftpd 2.3.4 running on PID 882.
3. **Correlation**: The engine queries the local ExploitDB. It finds a match: "vsftpd 2.3.4 - Backdoor Command Execution".
4. **Mitigation (The "Block")**:
    - The operator (or auto-policy) triggers mitigation.
    - IronClad writes PID 882 into the kernel block_map.

5. **Enforcement**:
    - An attacker sends the backdoor trigger to vsftpd.
    - The vulnerable process attempts to spawn /bin/sh.
    - The **IronClad eBPF Probe** intercepts the call inside the kernel.
    - It sees PID 882 in the map.
    - It denies the execution and submits an alert event to the Ring Buffer.
6. **Alerting**: The dashboard flashes a "QUARANTINED" alert log.

6. System Requirements

- **OS**: Linux (Ubuntu 20.04+, Debian 11+, Fedora 34+, Arch).
- **Kernel**: 5.7+ (Required for BPF LSM support).
- **Privileges**: Root (Required to load eBPF programs).
- **Dependencies**: clang, llvm, libbpf.

7. Future Roadmap

- **Network Filtering**: Integration with XDP (eXpress Data Path) to drop malicious packets at the NIC level before they reach the OS stack.
- **Container Awareness**: Namespace filtering to protect specific Docker/Kubernetes pods.
- **Heuristic Analysis**: eBPF-based anomaly detection for 0-day threats based on syscall patterns (e.g., "Why is the web server trying to read /etc/shadow?").

IronClad Kernel Guardian represents the shift from reactive security (cleaning up after a hack) to proactive, kernel-enforced immunity. By controlling the kernel, you control the battlefield.