STN-Labz Technical Whitepaper v1.0

Project: Sentinel-S1 — Deterministic Integrity for Orbital Edge Compute
Date: April 4, 2026
Status: TRL-5/6 (Ground-validated, simulation-tested; advancing toward flight validation)

---

1. Executive Summary

As orbital infrastructure evolves toward distributed compute platforms, a critical security gap has emerged. Conventional terrestrial security models—signature-based, cloud-dependent, and human-in-the-loop—are fundamentally incompatible with the realities of space: high latency, radiation-induced faults, and systems that are effectively locked for life.

STN Sentinel is a deterministic, host-based enforcement engine designed for autonomous operation in constrained and disconnected environments. It establishes **Sovereign Integrity** by enforcing a known-good system state and terminating unauthorized execution in real time—independent of ground connectivity.

Sentinel functions as a host-resident enforcement layer providing autonomous integrity assurance—effectively serving as an **immune system for orbital compute nodes**.

---

2. The Orbital Threat Matrix

The Sentinel Threat Network (STN) identifies three primary failure vectors for space-based compute systems:

2.1 Single-Event Upsets (SEU)

Radiation-induced bit flips in memory and registers can alter binary logic, corrupt execution paths, or destabilize system state without warning.

2.2 Logic Drift

Unauthorized or unintended modification of system configuration—such as routing tables (*nftables*) or DNS resolvers (*dnsmasq*)—can enable lateral movement across satellite mesh networks or degrade mission integrity.

2.3 The Connectivity Gap

Traditional endpoint detection and response (EDR) systems depend on continuous communication with centralized control infrastructure. In orbit, where communication may be intermittent, degraded, or denied, these systems fail by design.

---

3. Technical Architecture: The Sentinel Engine

Sentinel is implemented in **pure C** with zero external library dependencies, minimizing attack surface and ensuring portability across ARM64 and other embedded architectures. The design prioritizes determinism, transparency, and operational resilience.

3.1 Deterministic State Enforcement

Sentinel operates on a **hard-logic baseline**, rejecting probabilistic or heuristic-driven security models.

- **State Verification**
  Continuous validation of filesystem and system state against a locally stored, encrypted baseline (`state.db`).
- **Execution Guard**
  Process execution is intercepted at the syscall boundary. Binary hashes and execution context are validated against the defined **Rules of Engagement (ROE)** prior to execution.
- **The Kill-Stop Mechanism**
  Any deviation from the approved baseline—whether by PID, binary hash, or execution path—results in an immediate `SIGKILL`, preventing persistence or lateral propagation.

3.2 The Slog-Burst Protocol

Sentinel employs a custom telemetry mechanism optimized for constrained and intermittent communication environments.

- **Compact Alerting**
  Security events are encoded into fixed 192-byte packets.
- **Priority Queuing**
  Critical alerts are transmitted ahead of non-essential telemetry.
- **Loss-Tolerant Transmission**
  Designed for degraded or intermittent links (e.g., Swarm, Starlink IoT), ensuring delivery of high-value security signals under constrained bandwidth conditions.

3.3 Fault Tolerance & Recovery

Sentinel is engineered to operate under conditions where hardware reliability cannot be assumed.

- **Watchdog Integration**
  Hardware and software watchdogs ensure automatic recovery from system instability.
- **Read-Only Core Operation**
  Core system components operate from a protected, immutable state to prevent corruption.
- **State Rehydration**
  Baseline integrity can be restored from verified sources following fault detection.

- **Safe-Mode Enforcement**
  In the event of critical anomalies, Sentinel can enforce a minimal operational state, preserving system integrity over functionality.

---

4. Mission Roadmap: From Labz to Orbit

Phase I — The Gateway

Deployment of Sentinel within hardened Linux-based edge systems (e.g., nftables/dnsmasq routers) to establish terrestrial validation and ground-gate enforcement.

Phase II — The Simulation

Controlled fault-injection and adversarial testing within the STN-Labz simulation environment, including:

- Radiation-induced bit-flip emulation
- “Living off the Land” (LOTL) attack scenarios
- System degradation and recovery validation

Phase III — The Launch

Deployment of the **Sentinel-S1 CubeSat payload** to establish flight heritage and validate deterministic enforcement under true orbital conditions (~17,000 mph, LEO environment).

---

5. The Sovereign Mandate

In orbital environments, there are no patch cycles, no live debugging sessions, and no guaranteed recovery paths. Systems must operate with absolute certainty, even in isolation.

Sentinel enforces a **known-good state with deterministic precision**, ensuring that compute nodes remain trustworthy regardless of environmental or adversarial conditions.

This is not adaptive security.
This is **enforced integrity**.

Closing Statement

STN-Labz is not developing conventional security software.
It is establishing the foundational control layer required for autonomous systems operating beyond Earth.

As humanity extends its infrastructure into orbit and beyond, **trust must be engineered—not assumed**.

Sentinel is that enforcement layer.