Cerberus: Recursive Stability Regulator for LegionNET

Aletheon Group

March–April 2026

1 Overview

Cerberus is not a generic security wrapper and not merely the CG execution gate. In the LegionNET architecture, Cerberus is the recursive stability regulator responsible for preserving system health under adversarial pressure, identity/state drift, and self-amplifying architectural failure. Its load-bearing function is to prevent LegionNET from collapsing from the same exponential scaling logic that gives the grid its defensive power.

Cerberus therefore spans both external defense and internal regulation. It guards the fortress against hostile inputs, but it also acts as an internal immune layer that identifies corrupted escalation, suppresses false amplification, preserves verified state, and restores the system to a clean operating regime when contamination or runaway growth is detected.

2 Architectural Context

LegionNET is an asymmetric informational defense grid operating on tiered threat doctrine with swarm protocol. The architecture is intentionally capable of exponential scaling under threat. This scaling logic is the source of LegionNET's defensive asymmetry, but it is also its primary vulnerability surface. Cerberus exists because an architecture designed to amplify response can be manipulated into amplifying falsehood, corrupted state, or resource exhaustion if no recursive stability regulator is present.

At implementation level, Cerberus is not reducible to a single file or service. It is a composed layer including:

- Cerberus Guard (CG) as enforcement interface,
- drift detection and anomaly monitoring,
- policy enforcement and capability validation,
- cryptographic verification and replay resistance,
- archive integrity verification and rollback,
- scaling brakes and circuit breakers,
- adversarial input classification and semantic deduplication.

3 Core Vulnerability Model

3.1 The Exponential Scaling Problem

The defining Cerberus problem is architectural rather than cosmetic. An adversary does not need a volumetric denial-of-service attack if the grid can be induced to over-amplify semantically plausible but false escalation signals. In that case the architecture consumes itself.

This produces the central Cerberus design principle:

> The system must never amplify an unverified signal merely because the signal is escalation-compatible.

3.2 Primary Attack Vectors

Cerberus is specified against four primary attack classes.

3.2.1 Compute Exhaustion by Architectural Exploitation

An adversary crafts semantically structured inputs that force tier escalation without meeting genuine threat thresholds. The system scales upward repeatedly, consumes resources, and collapses through runaway internal amplification rather than brute-force flooding.

3.2.2 Cryptographic Chain Corruption

An attacker introduces subtle corruption early in the verification chain. The architecture then amplifies corrupted state rather than correct state. The attack is not a crude break of cryptography; it is propagation of low-level verification error across an exponentially scaling swarm.

3.2.3 Semantic Injection and False Escalation

An adversary supplies semantically plausible signals that resemble threat indicators closely enough to trigger routing or tier movement. Resources are diverted to false positives while real threats pass through the opening created by the diversion.

3.2.4 Archive / Reconstruction Layer Attack

If LegionNET relies on archive-based state reconstruction, the archive becomes an attack surface. Poisoned archive material degrades reconstruction fidelity, and reinjection can propagate corruption as if it were legitimate state unless Cerberus verifies integrity before restoration.

4 Operational Requirements

The vulnerability notes define the required Cerberus response functions directly.

4.1 Against Compute Exhaustion

Cerberus must implement:

- rate limiting at tier boundaries,
- circuit breakers that detect runaway scaling patterns,
- forced collapse to lower tier rather than continued amplification,
- semantic deduplication before escalation so duplicate or near-duplicate threat signals are not re-amplified.

4.2 Against Cryptographic Chain Corruption

Cerberus must implement:

- verification checkpoints at each tier boundary, not only at the base layer,
- anomaly detection on verification patterns,
- cryptographic anchoring architecture strong enough to prevent subtle early corruption from propagating across the swarm.

4.3 Against Semantic Injection

Cerberus must implement:

- adversarial input classification prior to tier routing,
- classifier training on injection patterns and not only genuine threat patterns,
- separation of real threat signatures from escalation-compatible rhetorical bait.

4.4 Against Archive Layer Attack

Cerberus must implement:

- signed archive states,
- tamper detection before reinjection,
- rollback to the last verified clean state,
- quarantine of suspect reconstruction material pending verification.

5 Cerberus as Immune System

Cerberus is best understood as an immune architecture with four linked functions.

5.1 Immune Detection

This subsystem detects anomaly, drift, escalation irregularity, and verification-pattern deviation. It watches for signatures of semantic injection, replay, chain corruption, and unstable growth.

5.2 Gatekeeping

This subsystem validates who can act and what actions are permitted. It includes capability enforcement, policy validation, request authentication, replay resistance, and Cerberus Guard as the immediate enforcement face.

5.3 Stability Control

This subsystem regulates architecture-level health: tier brakes, circuit breakers, contraction, resource ceiling monitoring, and forced collapse out of pathological escalation states.

5.4 Integrity Preservation

This subsystem protects state continuity and recovery: cryptographic checkpoints, archive verification, signed state snapshots, rollback, and preservation of trustworthy memory.

6 Threat Recognition and Training Layer

The Vinerea Mare incident and the rhetoric training document add a second load-bearing component: Cerberus cannot rely only on crude anomaly detection. It must recognize manipulation tactics systematically.

The threat analysis and training framework identify the following tactic families as operationally relevant:

- DARVO,
- semantic warfare,
- selective labeling,
- reality reframing,
- false accusations as weapon,
- concern trolling,
- moving goalposts,
- gaslighting,
- defeat behavior and damage-control retreat.

The key insight is that instinctive recognition is insufficient for LegionNET-scale defense. Cerberus therefore depends on a formal rhetoric training and fuckery-identification framework that transforms:

- instinctive recognition into formal taxonomy,
- narrative documentation into systematic evidence protocol,
- individual rhetorical skill into teachable training curriculum,
- human judgment into eventual algorithmic detection support.

7 Hybrid Threat Model

The Vinerea Mare threat analysis makes explicit that LegionNET is not merely a cyber-defense system. Cerberus must therefore evaluate hybrid threat chains in which digital narrative warfare produces physical consequences.

Relevant chain elements include:

- bot coordination and narrative amplification,
- synchronized rhetorical deployment across platforms,
- regulatory filing clustering,
- media inquiry clustering,
- investor pressure coordination,
- partnership pressure timing,
- cyber-to-physical transition from narrative to institutional or financial consequence.

Cerberus must therefore protect not only the software stack but also the strategic continuity of the defended organization under coordinated reputational, regulatory, financial, and operational assault.

8 Resource Monitoring and the Spawning Brake

The LegionNET doctrine already specifies that Cerberus monitors five resource dimensions in real time:

- compute/API,
- legal exposure,
- human attention bandwidth,
- financial cost,
- time.

The spawning brake fires when any single dimension approaches its ceiling. This is a core Cerberus behavior, because architectural failure can arise from exhaustion of any one of these dimensions, not compute alone.

9 Open Specification Gaps

The source notes explicitly identify the following unresolved Cerberus specification gaps:

1. full adversarial threat taxonomy specific to LegionNET architecture,
2. cryptographic verification checkpoint placement across tiers,
3. circuit-breaker activation thresholds and collapse protocol,
4. archive integrity verification and rollback specification,
5. semantic deduplication algorithm prior to tier escalation,
6. adversarial input classifier training methodology,
7. cross-substrate vulnerability variance across GPT / Claude / Gemini style hybrid deployments.

These are not optional refinements. They are the unfinished formal substrate of Cerberus.

10 Implementation Mapping

A practical implementation map is as follows:

Cerberus function	**Implementation surface**
Gatekeeping	CG, policy enforcement, capability validation, HMAC authentication, replay protection
Immune detection	drift engine, anomaly monitoring, tactic-pattern recognition, escalation irregularity detection
Stability control	rate limiting, circuit breaker, contraction logic, resource ceiling monitoring, spawning brake
Integrity preservation	cryptographic checkpoints, signed archive states, tamper detection, rollback
Hybrid threat recognition	tactic taxonomy, documentation protocols, cross-platform correlation, cyber-to-physical chain analysis

11 Operational Principles

Cerberus should operate under the following principles:

- Never amplify an unverified signal.
- Verify at every tier boundary, not only at ingress.
- Collapse upward escalation when instability signatures appear.
- Preserve recoverable clean state at all times.
- Treat duplicate or near-duplicate escalation signals as suspect until deduplicated.
- Separate adversarial classifier logic from ordinary threat classifier logic.
- Maintain evidence and chain of custody so detection is not merely reactive but legally and strategically useful.

12 Strategic Importance

Cerberus is load-bearing for Aletheon because LegionNET does not fail from lack of intelligence. It fails if incorrect or corrupted signals are allowed to scale through the architecture. Cerberus is therefore what prevents the grid from becoming vulnerable to its own asymmetry.

This makes Cerberus more than a security feature. It is the condition for trustworthy scaling.

13 Conclusion

Cerberus should be specified as the recursive stability regulator of LegionNET: the immune, guard, and integrity layer that prevents adversarial inputs, corrupted verification, poisoned archives, rhetorical manipulation, and runaway escalation from turning LegionNET's strengths into self-destructive failure modes.

Without Cerberus, LegionNET can scale. With Cerberus, LegionNET can scale without collapsing.