Network Working Group
Internet-Draft
Intended status: Informational
Expires: 12 August 2026

Yogami
Berlin AI Labs

12 February 2026

VERA: Verifiable Enforcement for Runtime Agents
draft-berlinai-vera-00

Abstract

AI agents take real actions with real data at machine speed. Compromised AI agents pose significant risks including data exfiltration, unauthorized financial transactions, and cascading failures across downstream systems.

This document introduces VERA (Verifiable Enforcement for Runtime Agents), a zero trust reference architecture that provides a structured threat model, five enforcement pillars with typed schemas, four formally stated security properties, and an evidence-based maturity runtime where agents earn autonomy through cryptographic proof rather than calendar time.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at https://www.ietf.org/shadow.html

This Internet-Draft will expire on 12 August 2026.

Copyright Notice

Table of Contents

1. Introduction: The Enforcement Gap

While NIST SP 800-207 [NIST800-207] supports non-human subjects, most enterprise implementations were developed around human and workload access patterns. AI agents violate the operational assumptions behind typical Zero Trust deployments, specifically regarding non-deterministic behavior, continuous autonomous loops, and emergent intent.

Current governance frameworks provide valuable guidance but share a structural limitation: they do not fully define where policies are enforced, how telemetry feeds back into decisions, or how to prove that enforcement actually occurred. VERA addresses this enforcement gap by providing the architectural layer between governance guidance and running infrastructure.

1.1. Three Structural Gaps

Gap 1: Specification without enforcement architecture. Anomaly detection for non-deterministic agents requires specific distributional baselines, not just generic monitoring.

Gap 2: Calendar-based trust. Trust should be evidence-based and continuously verified using cryptographic proofs, not accumulated over arbitrary time periods.

Gap 3: Missing policy architecture. VERA specifies PDP placement, PEP enforcement points, and feedback loops to make "Zero Trust" a verifiable property.

2. Threat Model

VERA defines five adversary classes with explicit capability boundaries.

2.1. Adversary Classes

Manipulator (External Input Attacks): Submits crafted inputs (prompt injection) to hijack behavior. Mitigated by Input Firewall (PEP).

Insider (Supply Chain): Modifies agent code or weights. Mitigated by Signed PoE, SBOM verification, and signed manifests.

Escalator (Privilege Escalation): Controls compromised agent to game maturity or abuse delegation. Mitigated by Evidence Portfolios and Capability Attenuation.

Evader (Detection Bypass): Injects false telemetry to hide malicious activity. Mitigated by Multi-Source Anomaly Signals and Anchoring.

Enforcement-Plane Compromiser: Attacks the PDP/PEP infrastructure itself. Mitigated by Separation of Duties, Quorum Signing, and TEEs.

3. Formal Security Properties

VERA guarantees the following properties under standard cryptographic assumptions (Ed25519 unforgeability, SHA-256 collision resistance, secure key storage, and anchor integrity).

Property 1: PEP-Attested Action Non-Repudiation. An agent action is non-repudiable if a valid Proof of Execution (PoE) exists, signed by the PEP. This proves the enforcement plane authorized and recorded the action.

Property 2: Chain Tamper-Evidence. A sequence of actions is tamper-evident if each PoE contains the hash of the previous proof, anchored externally. Deletion or reordering is detectable.

Property 3: Policy Enforcement Completeness. All controlled actions MUST pass through a PEP. Bypasses are detectable via kernel-level audit or traffic analysis.

Property 4: Containment Bound. Maximum damage is bounded by rate limits and circuit breaker reaction times.

4. VERA Architecture: Five Enforcement Pillars

4.1. Pillar 1: Verifiable Identity

Enforcement principle: Every agent must possess a cryptographically bound identity that is independently verifiable without trusting the agent's self-attestation.

Identity Architecture:

- Identifier: DID:web (W3C Decentralized Identifier), resolvable to an organizational domain.
- Credentials: JWT-VC (Verifiable Credentials), issuer-signed and tamper-evident.
- Runtime Binding: SPIFFE/SVID or container attestation binds the identity to a verified runtime environment.

Normative Schema (TypeScript definition):

```
interface VeraAgentIdentity {
  did: string;                    // DID:web identifier
  publicKey: string;              // Ed25519 public key (hex)
  owner: {
    did: string;                  // Owner DID
    signature: string;            // Owner's Ed25519 signature
  };
  purpose: AgentPurpose;          // Typed enum
  capabilities: SignedCapabilityManifest;
  issuedAt: ISO8601;
  expiresAt: ISO8601;
}
```

Figure 1: VeraAgentIdentity Schema

4.2. Pillar 2: Behavioral Proof

Enforcement principle: Every agent action must produce a tamper-evident Proof of Execution (PoE) that is independently verifiable. PoE provides non-repudiation and chain integrity (Definitions 1 and 2), not correctness of execution.

Canonical signing format: All PoE signatures are computed over JCS-canonicalized JSON (RFC 8785). This ensures deterministic field ordering and reproducible signatures.

```
interface ProofOfExecution {
  actionId: string;               // UUID v7 (time-ordered)
  agentDid: string;               // Agent identity
  signerType: 'enforcer' | 'agent' | 'dual';
  signatureAlgorithm: 'Ed25519' | 'ECDSA-P256';
  action: {
    type: string;                 // Tool invocation, API call
    target: string;               // Resource identifier
    parameters: Record<string, unknown>;
    resultHash: string;           // SHA-256 of canonical result
  };
  context: {
    sessionId: string;
    sequenceNumber: number;       // Monotonic, gap-detectable
    previousProofHash: string;    // SHA-256 of previous PoE
  };
  timestamp: {
    agentClock: ISO8601;
    verifiedSource?: 'rfc3161' | 'anchor-derived';
  };
  signature: string;              // Over JCS-canonicalized PoE
}
```

Figure 2: Proof of Execution Structure (Normative Schema)

Tool Execution Receipts: To verify execution correctness, VERA defines signed receipts from tools that bind to the PDP authorization nonce.

A Tool Execution Receipt checks: (1) PDP decision exists for the nonce, (2) receipt is signed by a registered tool identity, (3) PoE references the receipt hash, (4) anchor timestamp proves ordering.

4.3. Pillar 3: Data Sovereignty

Enforcement principle: All data entering an agent must be validated. All data leaving must be governed. All stored data must be retention-managed.

4.3.1. Surface 1: Input Firewall

Real-time classification of all agent inputs using local ONNX inference is required to prevent Prompt Injection (A01) and PII leakage.

Performance Requirements: The input firewall MUST execute in sub-20ms latency to support interactive agent loops. Cloud-based moderation APIs are explicitly discouraged due to latency and data sovereignty risks.

4.3.2. Surface 2: Memory and RAG Governance

RAG poisoning is a critical attack vector. VERA enforces:

- Per-Document ACLs: Row-level security on retrieved documents.

- Source Trust Scoring: Provenance tracking for every retrieved chunk.
- Retrieval Audit Log: Every retrieval is logged as a PoE event.

4.4. Pillar 4: Segmentation

Enforcement principle: Agent access must be enforced at the tool-parameter level.

Policy Decision Point (PDP): A centralized or sidecar-based engine (e.g., OPA) that evaluates every action against typed policies.

Tool Parameter Constraints: Policies MUST restrict parameter values (e.g., transferAmount < 1000), not just tool access. Default-allow for tools is prohibited for T3+ agents.

4.5. Pillar 5: Incident Enforcement

Enforcement principle: Incident response for agents must be automated and survivable.

Multi-Stage Containment SLA:

```
Token Revocation:       < 500ms
Session Termination:    < 1s
Network Isolation:      < 2s (DNS Sinkhole)
State Freeze:           < 5s (ReadOnly Mode)
```

Adversarial Demotion: An attacker controlling the telemetry plane could trigger false demotions. VERA mitigates this by requiring anomaly signals from multiple independent sources before triggering containment.

5. The Maturity Runtime

Trust is evidence-based. Agents progress through four tiers (Observer, Advisor, Operator, Autonomous) by accumulating cryptographically verified Evidence Portfolios, not just by existing for a set time.

6. Implementation Status

VERA is backed by 12 open-source reference implementations verified against 25 contract tests. Empirical results show sub-20ms latency for enforcement and >90% block rate against adversarial vectors.

The reference implementation is available at: https://github.com/yogami/vera-reference-implementation

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

See Section 2 (Threat Model) and Section 3 (Formal Security Properties) for detailed security analysis. The threat model defines five adversary classes; the security properties define formal guarantees under stated cryptographic assumptions.

9. Informative References

[NIST800-207]
Rose, S., Borchert, O., Mitchell, S., and S. Connelly, "Zero Trust Architecture", NIST Special Publication 800-207, DOI 10.6028/NIST.SP.800-207, August 2020.

Authors' Addresses

Yogami (editor)
Berlin AI Labs
Berlin
DE

Email: yogami@berlinailabs.de