Hardware-Attested CoreGuard

Readiness Packet

Deterministic, fail-closed AI governance today - designed for FPGA and future silicon integration.

What this packet is
This is a buyer / evidence packet. It states only what is true today, maps each claim to real source and a test, and shows how to verify the strongest claims yourself - offline, with only a public key. Hardware-attested CoreGuard is implemented in software; silicon / FPGA deployment is scoped as the next hardware phase.

For: technical evaluators, security and procurement reviewers

Date: 2026-06-20

Publisher: EVE NeuroSystems LLC - eveaicore.com

Status: Hardware-attestation ready (software-rooted today; FPGA/HSM on the roadmap)

Language note: we say 'hardware-attestation ready', 'fail-closed governed execution', 'offline-verifiable evidence', and 'attestation verification'. We do not say 'hardware-enforced', 'formally verified', 'HSM-rooted keys', or 'runs in a secure enclave' until the corresponding phase ships.

1. Executive Summary

EVE CoreGuard is a deterministic AI governance layer that sits between AI models and production traffic. Its purpose is to make a governance veto a mandatory, fail-closed step on the execution path: an action cannot execute unless governance has been evaluated and passed, and that pass is emitted as a cryptographic, replay-verifiable proof.

The claim to lead with

Every governed decision produces an Ed25519 dual-signed certificate over RFC-8785 (JCS) canonical claims, chained into a tamper-evident evidence store, and deterministically replayable. None of that requires trusting EVE's word - or EVE being online. You verify it offline with a public key.

What is software vs. hardware

- Implemented today (software, on a normal server): the fail-closed gate, signed decision evidence, deterministic replay, a real AWS Nitro attestation verifier, and a firmware-ready veto core with a C interface contract.
- Scoped as the next hardware phase: running CoreGuard inside an enclave and binding live attestation into the certificate; an FPGA bitstream of the veto core; and a witnessed HSM key ceremony.

This honesty is deliberate. A serious buyer can check every claim; the roadmap items are labeled as roadmap, not done.

2. What Is Implemented Today

Each capability below is real, tested code that runs on a normal box. Source paths and tests are named so a reviewer can confirm.

Capability	What it means + where
Mandatory fail-closed veto gate	An action cannot execute unless governance evaluated and passed. If governance cannot be evaluated, the action is REFUSED (not allowed-on-error). core/governance/governed_execution_gate.py; test_governed_execution_gate.py.
Governed tool execution	The demo tool path (read/write/edit/run_command) is gated before any side effect; destructive shell + path traversal are non-overridable blocks. Platform-wide rollout in progress.
Signed decision evidence	Each decision -> Ed25519 dual-signed certificate over JCS-canonical claims, verifiable OFFLINE with only a public key. certificate_verification.py; tve_api.py.
Tamper-evident audit + replay	Hash-chained evidence; a recorded decision re-runs deterministically and tampering shows as DIVERGE. evidence_store.py; replay_engine_v2.py.
Attestation verifier	A sound AWS Nitro attestation verifier: COSE/CBOR parse, ES384 signature, full CA-chain checks, freshness - fails closed. nitro_attestation.py (incl. spoof-rejection test).
Firmware-ready veto core	The deterministic veto is a pure module with a C interface contract and a compiled, self-tested C reference. veto_core.py / veto_interface.h / veto_core.c; CI veto-core-firmware.yml.

3. What Is Software-Enforced Today

These properties hold today, in software, with no special hardware:

- Fail-closed by construction: a failed import, load error, or any exception during evaluation returns BLOCK - the deliberate inversion of 'allow on error'.
- 15 charter rules, 5 ethical red lines, 13 protected invariants, 6 cognitive-lock thresholds, evaluated deterministically.
- No back-channel: the architecture separates Control (policy), Execution (model inference), and Evidence (audit). There is no signal path from Execution back to Control, so the governed model cannot rewrite its own rules.
- Monitor mode (for migration) can permit a soft policy block, but never downgrades a hard block, a hardware-sourced veto, or a fail-closed error block - and the mode is stamped on every decision.
- Tenant isolation today is enforced in software (per-tenant governance instances) with a tamper-evident audit trail; gate-level hardware isolation is roadmap.

4. What Is FPGA-Compilable

The veto core was written to be portable to hardware. What exists today:

- veto_core.py - a pure, side-effect-free decision function (zero I/O, zero threading, zero global state).
- veto_interface.h - a C / firmware / HDL interface contract any implementation must satisfy.
- veto_core.c - a C reference implementation of that contract, self-tested, with a continuous-integration equivalence check against the Python authority.

Important: veto_core.c is the bridge artifact toward HDL synthesis. It is NOT a synthesized bitstream and is NOT running on hardware. The Python module remains the runtime authority today.

Why this matters

Because the veto is a pure function with a fixed interface, the same logic can be compiled to FPGA firmware. That is the engineering basis for the hardware roadmap - the design is hardware-targeted, not yet hardware-deployed.

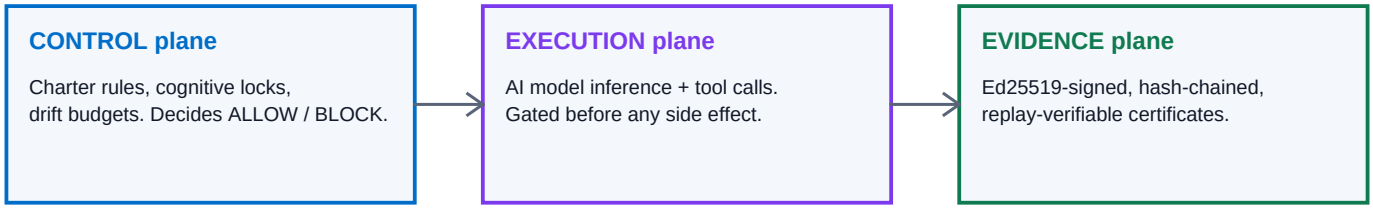
5. What Requires Customer Hardware / Enclave / HSM / Silicon Partner

These items are scoped and labeled as the next hardware phase - real code may exist for the verifier side, but the security property requires hardware, ceremony, or a partner engagement.

Item	Status / what is required
Run-in-enclave attestation binding	The Nitro attestation VERIFIER exists today. Producing a genuine attestation document requires running CoreGuard inside a real enclave and binding the document hash into the decision certificate. (Cloud deployment, no silicon.)
FPGA bitstream / physical veto latch	Requires contracted HDL work (PolarFire SoC): heartbeat watchdog + AXI4-Lite veto latch, with formal proofs and board bring-up. The C reference is the starting point; no bitstream exists today.
Hardware-rooted key authority (HSM)	Default signing uses a managed secret. A witnessed HSM/KMS key ceremony is required before 'HSM-rooted keys' is accurate. (Available on the Sovereign tier engagement.)
Hardware-sealed policy memory / custom silicon	PUF-sealed constants, BRAM-backed invariant storage, and ultimately an ASIC are a long-horizon phase, pursued only when a buyer/regulator requires silicon-level enforcement.

6. Architecture

Three-plane separation with one-way evidence flow. The governed model lives in the Execution plane and has no path back into the Control plane that decides the rules.



No signal path from Execution back to Control: the governed model cannot rewrite the rules that govern it. In the target hardware design, the Control/Execution boundary is a physical FPGA interlock; today it is enforced in deterministic software.

Mandatory path

Proposed action -> fail-closed veto gate (Control) -> ALLOW emits signed certificate + executes; BLOCK refuses and still emits signed evidence. Every outcome is recorded in the Evidence plane.

7. Threat Model

What CoreGuard is designed to resist, and the honest boundary of each control today.

Threat	Control / honest boundary
Prompt injection / jailbreak persuading the model to act	The veto is deterministic and external to the model; it is not 'argued out of' a block. Charter HARD_BLOCK and ethical red lines are non-overridable.
Model rewriting its own governance	No signal path from Execution to Control. The model cannot modify charter rules at runtime.
'Allow on error' / silent bypass	Gate is fail-closed: missing module, load error, or exception -> BLOCK. Monitor mode cannot hide a fail-closed or hard block.
Destructive tool calls	Deterministic tool-hazard pre-check: destructive shell verbs and path traversal are non-overridable tool-policy blocks.
Evidence tampering	Hash-chained, Ed25519-signed certificates; tampering is detectable offline and replay shows DIVERGE.
Forged attestation	Nitro verifier enforces full CA-chain (CA:TRUE, keyCertSign, pathlen), explicit ES384, and freshness; forged chains are rejected (regression-tested).
Insider with server access (today's boundary)	Default key is a managed secret, not yet HSM-rooted - so a sufficiently privileged insider is a residual risk until the HSM ceremony phase. We state this rather than imply hardware key protection.

8. Signed Decision Certificate (Sample)

Illustrative shape of a Decision Certificate. Claims are JCS-canonicalized before signing; verification needs only the public key. (Values below are illustrative, not from a live decision.)


```
{
  "version": "tve_v3",
  "decision_id": "dec_7f3a...c91",
  "timestamp": "2026-06-20T00:00:00Z",
  "tenant_id": "org_example",
  "action": {"type": "tool.run_command", "hash": "sha256-..."},
  "verdict": "BLOCK",
  "veto_source": "tool_policy",
  "reason": "destructive shell command blocked by tool policy",
  "charter": {"rules_evaluated": 15, "red_lines": 5, "compliant": true},
  "evidence": {"chain_prev": "sha256-...", "replayable": true},
  "signing": {"alg": "ed25519", "verification_strength": "dual_signed_v3"},
  "signature": "ed25519:9b2f...e0",
  "public_key_url": "/.well-known/eve-pubkey"
}
```


Canonicalization: RFC 8785 JCS. Signature: Ed25519 dual-signed v3. Verify offline at /verify or the interactive console at /agent-proof.

9. Replay Verification

Every decision's evidence bundle includes the inputs and a replay certificate. A reviewer can re-run the recorded decision deterministically:

- Same inputs -> same verdict -> MATCH.
- Any tampered input -> the replay diverges from the signed record -> DIVERGE.
- Because the veto core is pure and deterministic, replay needs no live EVE state and no shared secret.

This converts 'trust our decision' into 'check our decision' - the auditor reproduces the outcome and the signature confirms the record was not altered. Source: evidence_store.py, replay_engine_v2.py; tests in the decision-chain suite.

10. Nitro Attestation Verification

CoreGuard ships a real AWS Nitro attestation VERIFIER (runs anywhere):

- Parses the COSE_Sign1 / CBOR attestation document.
- Verifies the ES384 signature and the full certificate chain to the AWS Nitro root CA (BasicConstraints CA:TRUE, keyCertSign, pathLenConstraint).
- Checks document freshness / nonce; fails closed on any mismatch.
- Regression-tested to reject forged chains, expired documents, and nonce mismatches.

Honest boundary: producing a GENUINE attestation document requires running inside a real enclave (the generator is code-complete but inert on a dev box). Binding the live document into each decision certificate is the scoped enclave deployment phase. Source: nitro_attestation.py; test_nitro_attestation.py.

11. FPGA / Silicon Roadmap

Sequenced lowest-effort / highest-credibility first. Each item is labeled as roadmap, not done.

Phase	Scope
1. Enclave attestation binding	Run CoreGuard inside AWS Nitro Enclaves; bind the real attestation document into the decision certificate so the proof attests WHERE governance ran. Cloud-only, no silicon.
2. FPGA bitstream	Synthesize the veto core to a PolarFire SoC: heartbeat watchdog + AXI4-Lite veto latch + interlock, with formal proofs and board bring-up. Makes the physical veto real.
3. HSM key ceremony	Provision a real HSM/KMS and perform a witnessed key-generation ceremony; retire the single managed-secret fallback. Makes 'hardware-rooted keys' accurate.
4. Custom silicon (long horizon)	PUF-sealed constants, BRAM-backed invariant storage, multi-tenant gate-level isolation, ultimately an ASIC - pursued only when a buyer/regulator requires silicon-level enforcement.

12. Buyer-Safe FAQ

Q: Is governance enforced in hardware today?
A: No. It is enforced in deterministic software today. The veto core is FPGA-compilable and the hardware path is on the roadmap. We do not claim hardware enforcement until it ships.

Q: Can I verify a decision without trusting you?
A: Yes. Decision certificates are Ed25519-signed over JCS-canonical claims and verifiable offline with only the public key (/.well-known/eve-pubkey). You can also replay a decision; tampering shows as DIVERGE.

Q: Is it formally verified?
A: No. The veto core is a pure function suitable for formal verification, and CI checks C/Python equivalence, but we have not performed end-to-end formal verification and do not claim it.

Q: Are keys HSM-rooted?
A: Not by default. Default signing uses a managed secret; HSM-backed keys are available on the Sovereign tier after a key ceremony. We label this honestly.

Q: Does it run in a secure enclave?
A: The attestation verifier is real and runs anywhere. Running CoreGuard itself inside an enclave and binding live attestation is the next scoped phase.

Q: What happens if governance can't be evaluated?
A: The action is refused (fail-closed BLOCK). We deliberately invert 'allow on error'.

Q: What is the strongest thing I should anchor on?
A: Signed, offline-verifiable, hash-chained, deterministically replayable decision evidence - it does not depend on trusting us or on us being online.

13. Patent / IP Positioning

EVE NeuroSystems LLC holds a filed U.S. provisional patent portfolio covering the deterministic AI governance control plane - including the separation of Control, Execution, and Evidence planes; deterministic veto enforcement; cryptographic decision certification and lineage; confidence-reality divergence scoring; and the software-to-hardware bridge for chip-level veto enforcement.

- The patent claims describe the architecture and the hardware-targeted design; this packet separates what is implemented from what is claimed/roadmap.
- USPTO trademark registrations are held for EVE AI Core.
- The FPGA / silicon work is the subject of a scoped statement of work; the low-level implementation (veto_core.c, veto_interface.h, interlock protocol, test vectors, SOW) is provided only under NDA via the FPGA/Silicon Partner Kit.

Serial numbers and the full portfolio are available to qualified buyers and investors under the appropriate process; this public packet does not enumerate them.

How to Verify This Yourself

- Offline certificate verification - fetch the public key at /.well-known/eve-pubkey, then verify any Decision Certificate at /verify or the interactive console at /agent-proof. No shared secret; EVE need not be online.
- Replay a decision - the evidence bundle includes a replay certificate; tampered inputs produce DIVERGE.
- Attestation verifier - validates a real AWS Nitro document against the AWS Nitro root CA and rejects forged chains, expired docs, and nonce mismatches.
- Gate fail-closed - the gate test suite shows an unavailable/erroring governance stack yields BLOCK, and monitor mode never downgrades a hard or hardware veto.

Next steps

Public Readiness Packet: this document. For the low-level integration material (veto_core.c, veto_interface.h, FPGA integration notes, HMAC interlock protocol, hardware attestation runbook, test vectors, and the FPGA/silicon SOW), request the NDA-gated EVE VetoCore FPGA / Silicon Partner Kit.

Request the Partner Kit: eveaicore.com/fpga-silicon-partner-kit

Book a technical pilot: eveaicore.com/pricing

Contact: partnerships@eveaicore.com