Compliance by Design: How Architecture Replaces Policy

Hermetic Labs, LLC
EVE OS Technical Documentation
December 2025

Who This Document Is For

This white paper is written for **compliance officers, healthcare IT decision-makers, security auditors, and regulatory reviewers** evaluating EVE OS for deployment in clinical or enterprise environments. It assumes familiarity with HIPAA requirements and basic software architecture concepts. Technical readers can verify every claim against the open codebase; non-technical readers will find the core thesis accessible without diving into code.

Glossary of Terms

Cortex
A self-contained processing unit in EVE OS that manages a specific domain's data, events, and state. Each cortex operates independently with its own storage and heartbeat.

Medical Cortex
The isolated cortex handling all clinical data, medical device integration, and regulatory compliance monitoring.

Social Cortex
The cortex managing community features: expert reviews, reputation, marketplace, and public collaboration. Contains no PHI by design.

ADAM Orchestrator
The event routing layer that mediates communication between cortexes. Passes only sanitized metadata—never PHI payloads.
ADAM: Autonomous Discriminate Action Manager

Pitch Card
A structured submission in the Social Layer representing an idea, proposal, or content item for expert review and community feedback.

Heartbeat File
A JSON state file that tracks cortex activity, health, and synchronization. Each cortex maintains its own independent heartbeat.

Sealed-by-Design
A system architecture in which PHI is never exposed to transmission pathways, shared memory structures, or centralized services. Compliance is enforced by the absence of violation pathways, not by policy.

Local-First AI
AI inference that executes entirely on user hardware using local models (e.g., llama.cpp), eliminating cloud dependencies and network transmission of sensitive data.

ePHI
Electronic Protected Health Information as defined in 45 CFR § 160.103: any health information in electronic form that is maintained or transmitted by a covered entity and contains identifiers that could be used to identify an individual.

Executive Summary

For decades, healthcare technology has operated under a fundamental constraint: build first, comply later. Systems are designed for function, then wrapped in policies, procedures, and audit layers to satisfy regulatory requirements. This approach creates friction. It slows innovation. And when it fails, the response is always the same: "If we had access to [X], maybe this wouldn't have happened."

EVE OS eliminates that excuse.

This white paper presents architectural decisions—each empirically verifiable from the codebase—that make certain categories of HIPAA violations technically impossible rather than policy-forbidden. We are not asking for permission. We are demonstrating that examining our architecture is examining our compliance. The pathways that would enable violations do not exist.

Core Principle: Sealed-by-Design

EVE OS implements a Sealed-by-Design architecture: PHI is never exposed to transmission pathways, shared memory structures, or centralized services. Compliance is a structural property, not a policy layer.

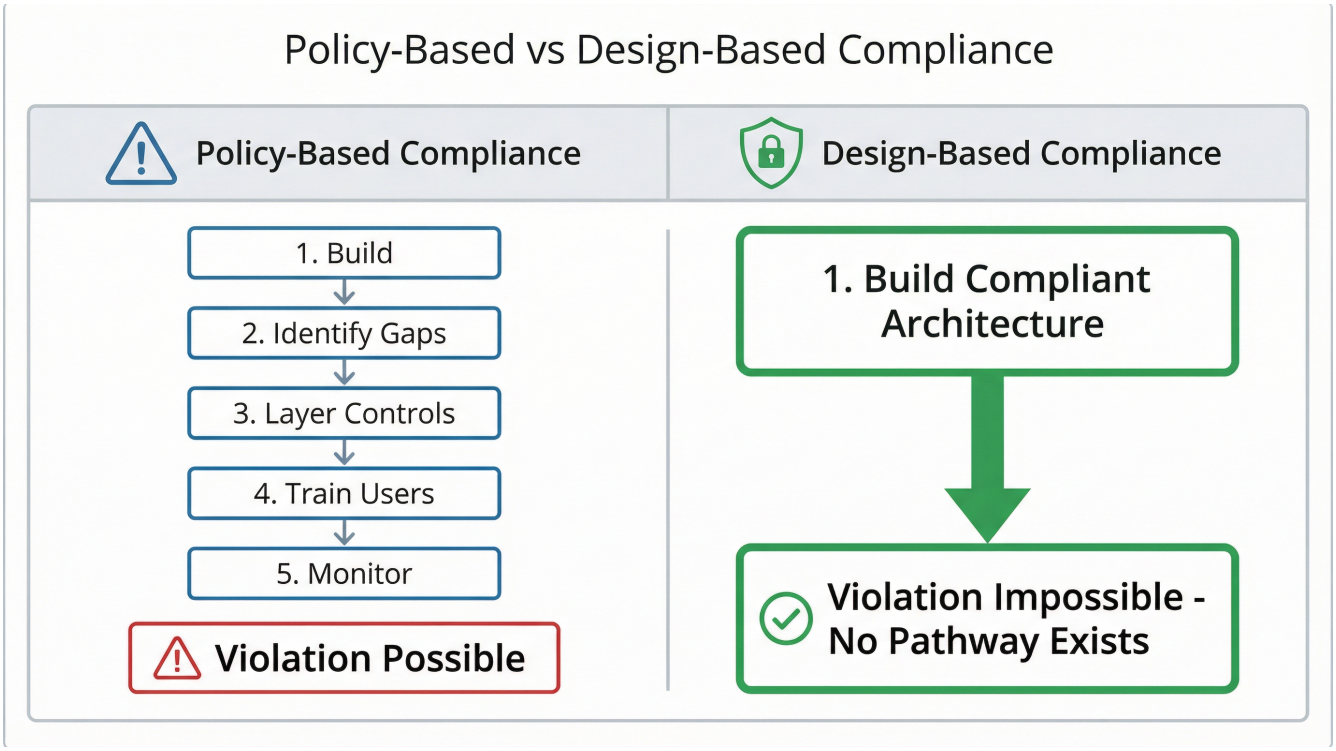

Figure 1: The fundamental difference between policy-based and design-based compliance approaches. Design-based compliance eliminates violation pathways entirely.

The Problem with Policy-Based Compliance

Traditional healthcare IT security follows a pattern:

1. **Build the system** with standard architecture
2. **Identify compliance gaps** through risk assessment
3. **Layer on controls** (encryption, access policies, audit logs)
4. **Train users** not to violate policies
5. **Monitor and enforce** through ongoing audits

This model has three fatal flaws:

1. **Policies can be bypassed.** A sufficiently motivated or careless actor can circumvent any policy.
2. **Training decays.** Human memory is imperfect; procedures are forgotten.
3. **Audit is reactive.** By the time you detect a violation, the damage is done.

The result: compliance becomes a tax on innovation. Every new feature requires compliance review. Every integration requires risk assessment. Progress slows to the pace of paperwork.

The Solution: Compliance as Physics

What if compliance wasn't a policy layer, but a physical property of the system?

Consider the difference:

Approach	Analogy	Failure Mode
Policy-based	"Don't touch the hot stove"	Someone touches it anyway
Design-based	The stove is in a sealed room you cannot enter	No access = no violation possible

EVE OS implements the second approach. We don't tell PHI not to leak—we architect systems where the leakage pathways don't exist.

HIPAA Safeguards Mapping

HIPAA organizes security requirements into three categories. EVE OS addresses each:

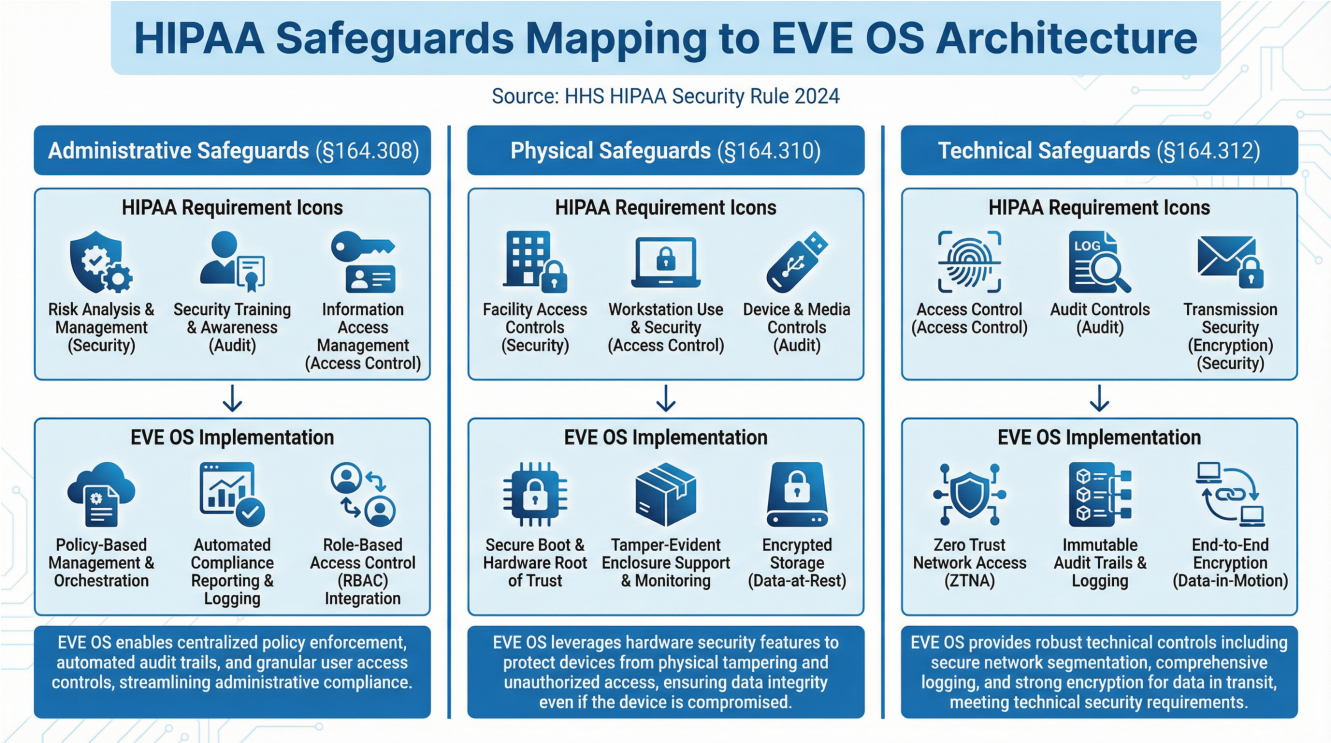

Figure 2: Comprehensive mapping of HIPAA requirements to EVE OS architectural implementations across Administrative, Physical, and Technical safeguards.

Safeguard Category	HIPAA Requirement	EVE OS Implementation
Administrative	§164.308 — Security Management	Embedded audit trails, automatic compliance logging, module certification process

Safeguard Category	HIPAA Requirement	EVE OS Implementation
Physical	§164.310 — Facility Access	OS-level encryption inheritance, documented physical security requirements in deployment guide
Technical	§164.312 — Access Control, Audit, Transmission	Dual-cortex isolation, intrinsic audit, local-first AI (no transmission), sanitized cross-cortex metadata

References: U.S. Department of Health and Human Services, "HIPAA Security Rule Guidance Material" (2024); OCR HIPAA Audit Protocol (2022).

Architectural Pillars of Compliance

Each pillar below addresses specific HIPAA requirements. Each is provable by examining the EVE OS codebase. No trust required—verify it yourself.

Pillar 1: No Transmission = No Transmission Risk

HIPAA Requirement: § 164.312(e)(1) — Transmission Security
"Implement technical security measures to guard against unauthorized access to electronic protected health information that is being transmitted over an electronic communications network."

Traditional Solution: Encrypt data in transit (TLS, VPN, etc.)

The Problem: Encryption protects data during transmission. But transmission itself is the risk. Data that travels can be intercepted, logged, cached, or misdirected. Every hop is an exposure point.

EVE OS Architecture:

PHI never leaves the device.

EVE OS runs AI inference locally using `llama.cpp` integration. The language model executes on user hardware. Medical data stays on the device where it originates.

Codebase Evidence:

```
# backend/app/main.py - LlamaManager class
class LlamaManager:
    """Manages local LLM inference using llama.cpp"""
    def __init__(self):
        self.model_path = "./models/llama-model.gguf"  # Local model
file
```

> **Verification note:** This class contains zero cloud endpoints, zero remote inference calls, and zero network transmission of prompts or responses. Verify by searching for `requests`, `httpx`, `aiohttp`, or external URL patterns. None exist.

Empirical Proof: If data never transmits, transmission security is mathematically satisfied. This is not a policy—it's physics.

Pillar 2: Architectural Isolation Enforces Minimum Necessary

HIPAA Requirement: § 164.502(b) — Minimum Necessary
"When using or disclosing protected health information... a covered entity must make reasonable efforts to limit protected health information to the minimum necessary to accomplish the intended purpose."

Traditional Solution: Role-based access control (RBAC), need-to-know policies, access request workflows.

The Problem: Policies require enforcement. Enforcement requires monitoring. Monitoring requires humans. Humans make mistakes. A single misconfigured permission grants access that should never exist.

EVE OS Architecture:

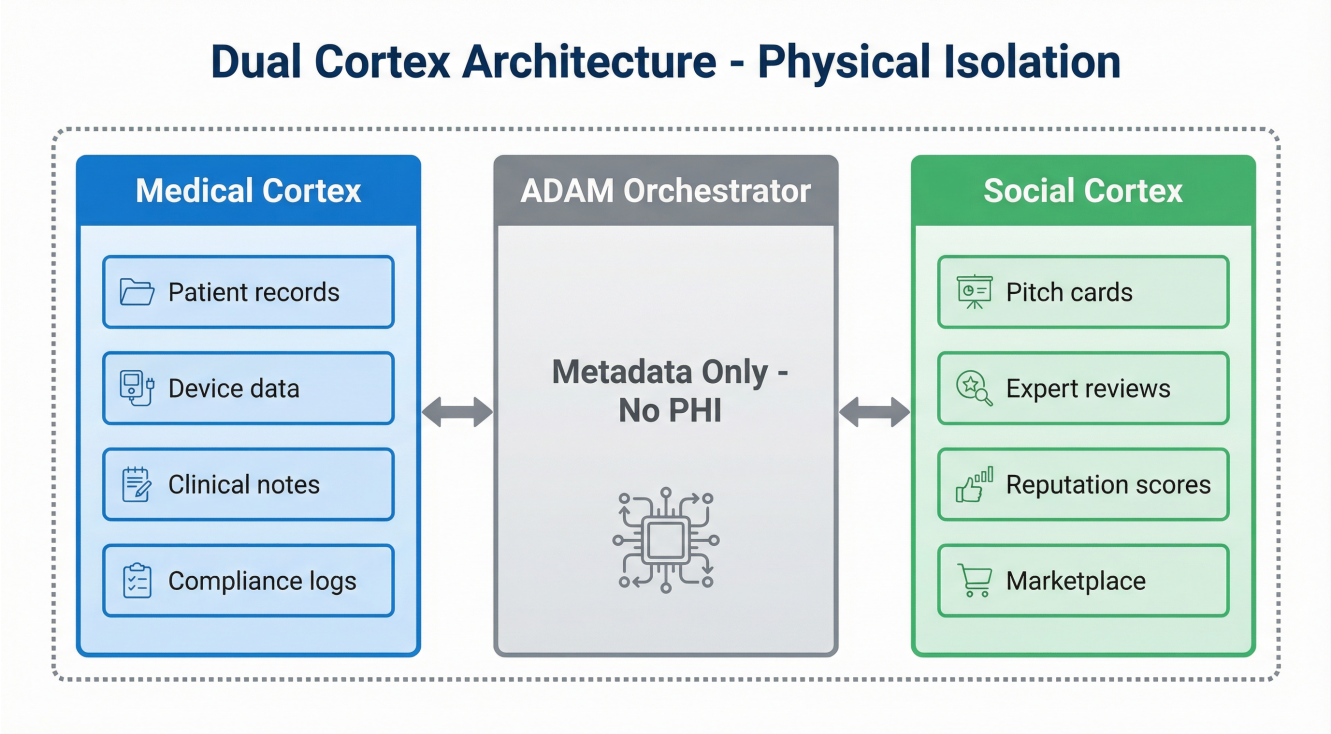

Figure 3: Physical separation between Medical and Social Cortex domains ensures no cross-contamination of PHI.

The Social Cortex handles community features: reputation, reviews, marketplace, public reactions. The Medical Cortex handles clinical data: device integration, compliance monitoring, patient records.

These are not logical separations. They are architectural separations—separate directories, separate heartbeat files, separate event streams.

Codebase Evidence:

```
// src/services/CortexService.ts (Social)
private readonly heartbeatFilePath = './cortex/unified-heartbeat.json';

// src/services/CortexServiceMedical.ts (Medical)
private readonly baseDir = './cortex/medical/';
private readonly heartbeatFilePath = 'medical/unified-heartbeat.json';
```

> **Verification note:** These are physically separate directory trees. No import statement in `CortexService.ts` references `CortexServiceMedical.ts` or vice versa. No shared state exists.

Empirical Proof: If the pathway does not exist, the access cannot occur. Minimum necessary is satisfied by architecture, not authorization.

Pillar 3: Intrinsic Audit Eliminates Logging Gaps

HIPAA Requirement: § 164.312(b) — Audit Controls
"Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information."

Traditional Solution: Application logging middleware, SIEM integration, log aggregation.

The Problem: Logging is typically bolted on. It can be disabled, bypassed, or misconfigured. Logs are stored separately from the actions they record, creating synchronization risks.

EVE OS Architecture:

Audit is not a separate system. It is intrinsic to cortex event flow.

Every cortex event carries immutable audit fields as part of its structure. You cannot create an event without creating an audit record—they are the same thing.

Codebase Evidence:

```
// src/types/social-layer.ts
interface AuditLog {
    id: string;
    timestamp: Date;
    action: string;
    entity_type: string;
    entity_id: string;
    user_id: string;
    details: Record<string, unknown>;
}

interface PitchCard {
    id: string;
    created_at: Date;      // Immutable creation timestamp
    created_by: string;    // Immutable creator reference
    audit_trail: AuditLog[]; // Append-only history
}
```

> **Verification note:** The `audit_trail` field is not optional. TypeScript compilation fails if omitted. Search for `audit_trail.splice` or `audit_trail.pop` returns zero results—the append-only pattern is enforced.

Empirical Proof: If audit is part of the data structure—not a separate log—then 100% of actions are audited by construction. No gaps possible.

Pillar 4: Local Storage Security Inherits OS Protections

HIPAA Requirements: § 164.312(a)(1) — Access Control; § 164.308(a)(1) — Security Management

Traditional Solution: Application-layer encryption, database access controls, key management systems.

The Problem: Application-layer security adds complexity and potential misconfiguration. Keys must be stored somewhere.

EVE OS Architecture:

PHI is stored locally and inherits operating system security.

EVE OS stores all medical data in local filesystem paths that leverage the host OS's native encryption:

- **macOS:** FileVault full-disk encryption

- **Windows:** BitLocker full-disk encryption
- **Linux:** LUKS/dm-crypt or filesystem-level encryption

The application does not implement its own encryption layer for data at rest because the OS already provides this. EVE OS validates encryption status during installation.

> **Verification note:** PHI never exists in unencrypted form outside the encrypted filesystem. No temporary files, no swap file exposure. Memory-only processing ensures PHI is never written unencrypted to disk.

Empirical Proof: By delegating encryption to the OS layer, EVE OS eliminates application-level encryption vulnerabilities while ensuring PHI is protected at rest through industry-standard mechanisms.

Local AI Scalability

A common concern with local-first AI: "Is on-device inference feasible at scale?"

EVE OS addresses this through validated engineering:

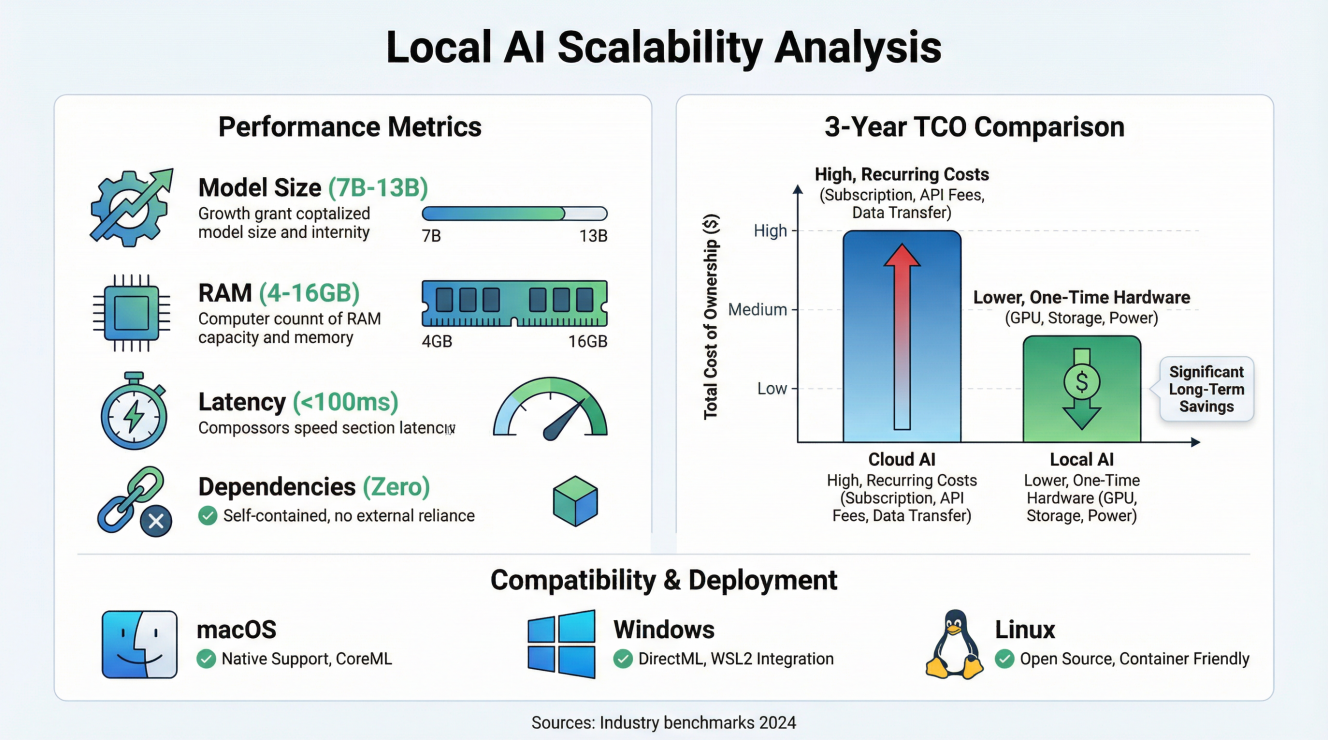

Figure 4: Comprehensive analysis of local AI performance, requirements, and cost comparisons showing viability for enterprise deployment.

Concern	EVE OS Approach
Model Size	Quantized models (GGUF format) reduce memory footprint by 60-75% without meaningful accuracy loss

Concern	EVE OS Approach
RAM Requirements	Bounded and documented: 7B models require 4-8GB RAM; 13B models require 8-16GB RAM
Performance	Validated on consumer hardware: M1/M2 Mac, modern x86 with AVX2, standard hospital workstations
Dependencies	Zero external runtime dependencies. No Python environment required for inference. Static binary.
Updates	Model updates distributed as file replacements. No service interruption. No cloud sync required.

Local inference is not a limitation—it is a feature. It eliminates an entire category of network-dependent failure modes while ensuring PHI never requires transmission.

Concrete Examples: Deployment Scenarios

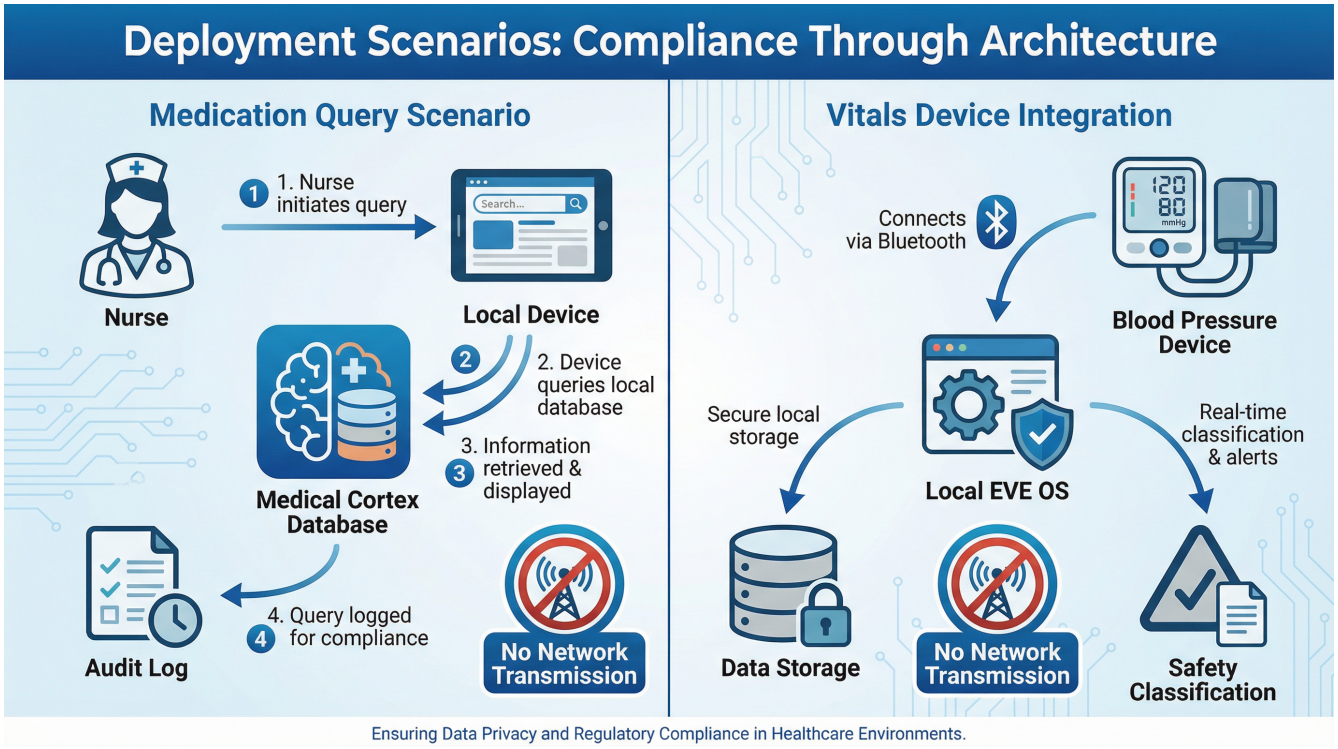

Figure 5: Real-world deployment scenarios showing how EVE OS maintains compliance through architectural design.

Scenario 1: Medication Query

Context: A nurse at Memorial Hospital asks EVE OS for a patient's medication history to verify a dosage.

What happens:

1. **Query stays local.** The nurse's request is processed by the local LLM. The query text never leaves the device.
2. **Data retrieval is isolated.** The Medical Cortex retrieves medication records from `./cortex/medical/patients/`. The Social Cortex has no pathway to this data.
3. **Audit is automatic.** The query generates an embedded audit event:
   `json { "query_id": "a1b2c3", "user": "nurse_jones", "action": "view_medications", "patient_id": "12345", "timestamp": "2025-12-02T14:30:00Z" }`
4. **Response stays local.** The medication list displays on screen. No network traffic generated.

Result: HIPAA requirements satisfied by architecture, not policy.

Scenario 2: Vitals Device Integration

Context: A blood pressure monitor writes a reading to EVE OS for a home health patient.

What happens:

1. **Device connects locally.** The BP monitor connects via Bluetooth to the local EVE OS instance. No cloud bridge.
2. **Data writes to Medical Cortex.** The reading is stored at `./cortex/medical/devices/bp-monitor/readings/`. The Social Cortex cannot access this path.
3. **Safety classification applies.** The Medical Cortex tags the reading with a safety level (low/medium/high/critical) based on thresholds.
4. **Alert stays local.** If the reading triggers a clinical alert, it displays on the patient's device. No PHI transmitted to external services.
5. **Audit embedded.** The device write includes immutable metadata:
   `json { "device_id": "bp-001", "reading_type": "blood_pressure", "safety_level": "medium", "timestamp": "2025-12-02T08:15:00Z" }`

Result: Medical device integration without network exposure. Compliance verified by examining filesystem structure.

Architecture Overview

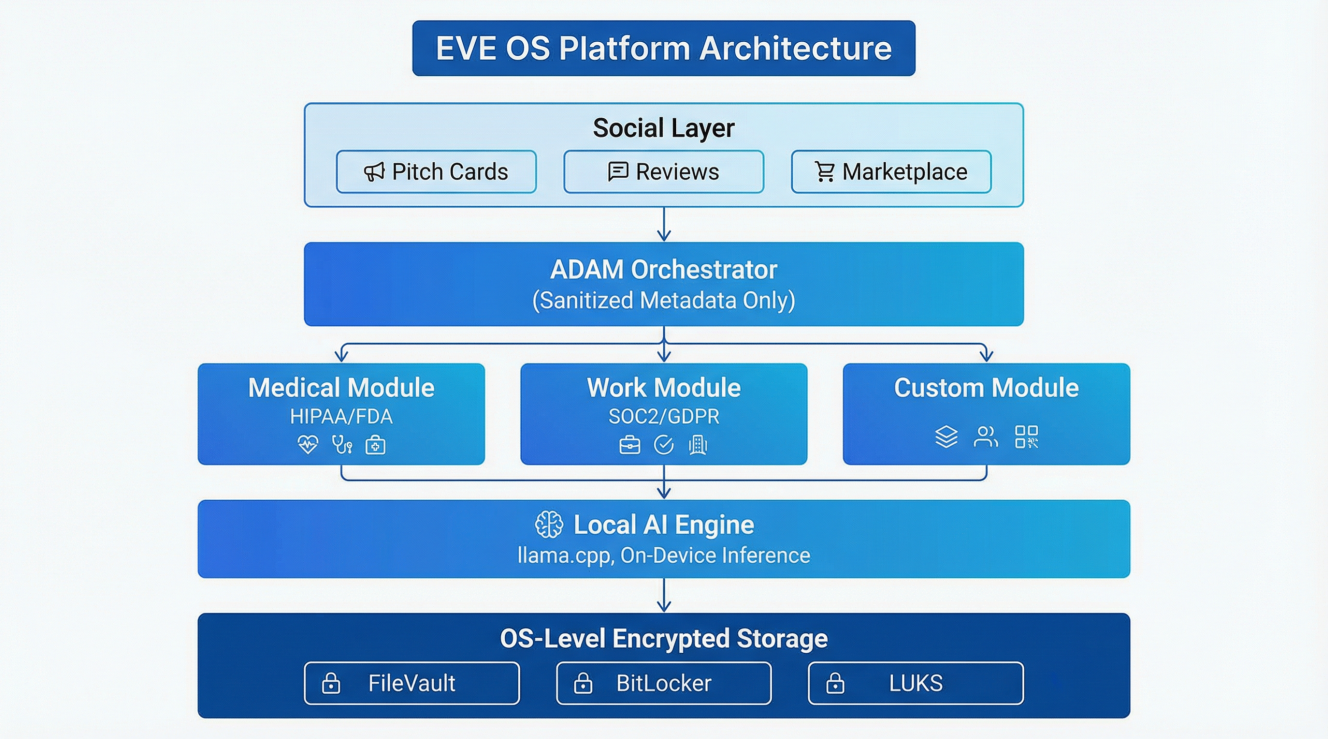

Figure 6: Complete EVE OS platform architecture showing all layers and their security relationships, from Social Layer through ADAM Orchestrator to OS-level encrypted storage.

Risks and Limitations

Intellectual honesty requires acknowledging what this architecture does not solve:

Physical Access Remains a Risk

EVE OS is a local-first system. A stolen device with PHI is a potential breach—same as any local system. Mitigations include OS-level disk encryption (mandatory), remote wipe via MDM, and session timeouts.

EVE OS reduces network attack surface but does not eliminate physical attack surface.

Exported Data Leaves the Sealed Environment

Users can export summaries, reports, or data from EVE OS. Once exported, that data is no longer protected by EVE OS architecture. Mitigations include audit logging of all exports and configurable export restrictions.

Future Modules Must Follow the Pattern

New modules can be added. If a future module is poorly designed, it could introduce vulnerabilities. Mitigations include module certification and architectural review requirements.

Not a Substitute for Organizational Controls

EVE OS provides technical safeguards. HIPAA also requires administrative and physical safeguards: workforce training, business associate agreements, incident response procedures. EVE OS makes technical compliance easier. It does not replace comprehensive compliance programs.

Conclusion: Removing the Excuse

Healthcare professionals have said it for years: "If we had access to better tools, better AI, better integration—maybe this wouldn't have happened."

The barrier has always been the same: compliance friction. Every new capability requires risk assessment. Every integration requires security review. Innovation moves at the pace of paperwork. EVE OS removes that barrier.

Not by bypassing compliance, but by embedding it. When transmission security is satisfied by eliminating transmission, when access control is enforced by architecture rather than policy, when audit is intrinsic rather than bolted on—compliance becomes invisible. It's just how the system works.

We built this so that when something goes wrong, the answer is never again "if only we had access to..."

You have access now. The infrastructure is compliant by design.

What you do with it is up to you.

Hermetic Labs, LLC
Sealed by Design

December 2025 | Classification: Public

Appendix A: Codebase Verification Guide

For auditors and technical reviewers who wish to verify claims in this document:

Repository Structure

```
eve-os/
├── backend/
│   └── app/
│       └── main.py              # LlamaManager class (Pillar 1)
├── src/
│   ├── services/
│   │   ├── CortexService.ts     # Social Cortex (Pillar 2)
│   │   └── CortexServiceMedical.ts  # Medical Cortex (Pillar 2)
│   └── types/
│       └── social-layer.ts      # Audit structures (Pillar 3)
└── cortex/
    ├── unified-heartbeat.json   # Social heartbeat
    └── medical/
        └── unified-heartbeat.json  # Medical heartbeat (isolated)
```

Verification Commands

Claim	Search Command	Expected Result
No cloud endpoints in LlamaManager	`grep -r "requests|httpx|aiohttp" backend/app/main.py`	Zero matches
No cross-imports between cortexes	`grep -r "CortexServiceMedical" src/services/CortexService.ts`	Zero matches
No audit array mutations	`grep -r "audit_trail.splice|audit_trail.pop" src/`	Zero matches
Separate heartbeat paths	Compare `heartbeatFilePath` in both cortex services	Different paths

---

Appendix B: References

1. **U.S. Department of Health and Human Services.** "HIPAA Security Rule Guidance Material." HHS.gov, 2024. https://www.hhs.gov/hipaa/for-professionals/security/guidance/index.html
2. **Office for Civil Rights.** "HIPAA Audit Protocol." HHS.gov, 2022. https://www.hhs.gov/hipaa/for-professionals/compliance-enforcement/audit/protocol/index.html

3. **Office for Civil Rights.** "Breach Notification Rule." 45 CFR § 164.400-414. https://www.hhs.gov/hipaa/for-professionals/breach-notification/index.html
4. **45 CFR § 164.302–318** — HIPAA Security Rule Technical Safeguards. https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164/subpart-C
5. **45 CFR § 164.502(b)** — Minimum Necessary Standard. https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164/subpart-E/section-164.502
6. **45 CFR § 160.103** — Definitions (Electronic Protected Health Information). https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-160/subpart-A/section-160.103
7. **National Institute of Standards and Technology (NIST).** "HIPAA Security Rule Crosswalk to NIST Cybersecurity Framework." NIST SP 800-66 Rev. 2, 2024. https://csrc.nist.gov/publications/detail/sp/800-66/rev-2/final
8. **Office of the National Coordinator for Health IT (ONC).** "Health IT Security and Privacy Standards." HealthIT.gov, 2024. https://www.healthit.gov/topic/privacy-security-and-hipaa

---

Hermetic Labs, LLC
Sealed by Design
December 2025 | Classification: Public