RTFCT

INFERENCE-TIME ENFORCEMENT

The architecture of active compliance versus passive monitoring

2026-07-28

ABSTRACT

AI compliance architectures fall into two postures. Passive Monitoring, implemented as a Sidecar, logs the input and output after the model has already executed the query, and relies on a later audit to detect violations. Active Enforcement, implemented as an Interceptor, evaluates the query against policy before it reaches the model, blocks or allows it in real time, and writes cryptographic evidence of the decision.

This paper compares the two architectures on data flow, latency, failure modes, evidentiary weight, and total cost, and sets out the migration path from sidecar logging to inference-time enforcement.

Technical White Paper for CTOs, VP Engineering, and CISOs

THE TWO ARCHITECTURES

OVERVIEW

AI compliance and security architectures fall into two fundamental postures: Passive Monitoring and Active Enforcement. The distinction lies in when and how compliance is evaluated relative to the AI inference process.

Passive Monitoring, typically implemented as a Sidecar, operates after the AI model has executed a query. It logs the input, output, and metadata for later review, relying on human or automated audits to detect violations. Active Enforcement, implemented as an Interceptor, operates before the query reaches the model. It evaluates the query against pre-defined policies and either blocks or allows it in real time, with cryptographic evidence of the decision.

ARCHITECTURAL DESCRIPTIONS

PASSIVE MONITORING (SIDECAR POSTURE)

- Data Flow:

Application sends query to AI model AI model executes query Sidecar captures input/output and logs to a database Audit process reviews logs for compliance violations.

- Latency Impact: Minimal to none, as logging occurs asynchronously after execution.
- Failure Modes:

- Violation is logged, but the non-compliant query has already been executed.
- Evidence is stored in alterable logs (e.g., database entries).
- Compliance gaps exist between execution and audit.

ACTIVE ENFORCEMENT (INTERCEPTOR POSTURE)

- Data Flow:

Application sends query to Interceptor Interceptor evaluates query against pre-compiled policies If compliant, Interceptor signs the query and forwards it to the AI model If non-compliant, Interceptor blocks the query and returns a cryptographic artifact proving the decision.

- Latency Impact: Real-time evaluation adds minimal overhead, optimized for instantaneous decision-making.
- Failure Modes:
- Query is either blocked or allowed with a tamper-proof cryptographic signature.
- No compliance gap: non-compliant queries are never executed.

ASCII DATA FLOW DESCRIPTIONS

PASSIVE MONITORING (SIDECAR POSTURE)

ACTIVE ENFORCEMENT (INTERCEPTOR POSTURE)

THE LATENCY MYTH

THE PERCEPTION OF OVERHEAD

A common objection to Active Enforcement is the assumption that real-time policy evaluation will introduce unacceptable latency. This concern stems from traditional rule-based systems where policies are interpreted at runtime, often in high-latency environments (e.g., cloud-based policy engines).

However, modern architectures optimize for performance through:

- Edge deployment: Policies are evaluated at the edge, close to the application.
- Policy pre-compilation: Policies are compiled into executable bytecode and loaded into the Interceptor at startup.
- Cryptographic signing at the gateway: Occurs at the edge, not in a remote service.

COMPARATIVE ROI

Scenario | Traditional Audit Cycle | Structural Compliance

Violation Detection Time | 90 days | Real-time

Compliance Gap | Yes | No

Evidence Integrity | Alterable | Tamper-proof

THE FAILURE MODE ANALYSIS

PASSIVE MONITORING: THE "DETECT AND REACT" PROBLEM

When Passive Monitoring fails, the consequences are severe:

- Violation Occurs: The non-compliant query is executed by the AI model.
- Damage Done: Sensitive data may be leaked, biased outputs generated, or regulatory violations committed.
- Evidence is After-the-Fact: Logs are written to a database, which can be altered or deleted.
- Compliance Gap: The time between execution and audit creates a window where violations go unnoticed.

ACTIVE ENFORCEMENT: THE "PREVENT AND PROVE" MODEL

When Active Enforcement fails, the system defaults to a secure state:

- Block by Default: If the Interceptor cannot evaluate a policy, it blocks the query and returns a cryptographic artifact proving the failure.
- Allow with Artifact: If the query is compliant, it is signed and forwarded. The signature serves as tamper-proof evidence of compliance.
- No Compliance Gap: Non-compliant queries are never executed.

FAILURE MODE COMPARISON

Failure Mode | Passive Monitoring (Sidecar Posture) | Active Enforcement (Interceptor Posture)

Query Executed? | Yes | No

Evidence Tamperable? | Yes (database logs) | No (cryptographic artifact)

Compliance Gap? | Yes | No

Default State | Allow | Block

POLICY AS CODE

THE PROBLEM WITH HUMAN-INTERPRETED POLICIES

Traditional compliance relies on:

- Documents: Policies written in natural language and interpreted by humans.
- Manual Audits: Engineers or auditors manually check logs against policies.
- Subjectivity: Interpretations of policies can vary, leading to inconsistent enforcement.

POLICY AS EXECUTABLE CODE

The Interceptor posture treats policies as executable rules, evaluated at inference time:

- Policy Language: Policies are written in a structured format for machine evaluation.
- Pre-Compilation: Policies are compiled and loaded into the Interceptor at startup.
- Runtime Evaluation: The Interceptor evaluates the compiled policy against each query before execution.

THE DIFFERENCE: "WE HAVE A POLICY" VS. "THE SYSTEM CANNOT EXECUTE A NON-COMPLIANT QUERY"

Approach | Human-Interpreted Policy | Policy as Code (Interceptor Posture)

Enforcement | Manual | Automatic

Consistency | Low (human error) | High (deterministic)

Speed | Slow (audits) | Real-time

Evidence | Logs | Cryptographic artifacts

Compliance Guarantee | No | Yes

CRYPTOGRAPHIC PROVENANCE

THE PROBLEM WITH DATABASE LOGS

Traditional logging systems store compliance data in databases, which have critical weaknesses:

- Alterable: Logs can be modified or deleted by administrators or attackers.
- Centralized: A single breach can compromise all logs.
- No Non-Repudiation: Cannot prove who created or modified a log entry.

CRYPTOGRAPHIC SIGNING AT THE EDGE

The Interceptor posture uses cryptographic signing to create tamper-proof evidence:

- Signing Process:
- The Interceptor generates a hash of the query, policy evaluation result, and timestamp.
- The hash is signed with the Interceptor's private key.
- The signed artifact is returned to the application and stored in an immutable log.
- Verification Process:
- Any party can verify the artifact using the Interceptor's public key.
- The signature proves the artifact was created by the Interceptor and has not been tampered with.

COMPARISON: DATABASE LOGS VS. CRYPTOGRAPHIC ARTIFACTS

Feature | Database Logs | Cryptographic Artifacts

Tamper-Proof | No | Yes

Non-Repudiation | No | Yes

Centralized | Yes | No (distributed)

Verifiable | No (trust required) | Yes (mathematical proof)

THE SIDECAR VS. INTERCEPTOR DUALITY

WHEN TO USE SIDECAR (PASSIVE MONITORING POSTURE)

The Sidecar posture is suitable for:

- Non-Regulated Use Cases: Where compliance is not a legal requirement (e.g., internal R&D, non-sensitive data).
- Debugging and Analytics: Capturing input/output for model improvement or debugging.
- Legacy Systems: Where modifying the application to integrate an Interceptor is not feasible.

WHEN TO USE INTERCEPTOR (ACTIVE ENFORCEMENT POSTURE)

The Interceptor posture is required for:

- Regulated Industries: Healthcare (HIPAA), finance (GLBA), or government (FedRAMP).
- Sensitive Data: PII, PHI, or proprietary information.
- High-Risk Queries: Queries that could cause harm if executed (e.g., biased outputs, data leaks).

THE UPGRADE PATH

Organizations can start with a Sidecar posture for non-critical use cases and upgrade to an Interceptor posture as compliance requirements evolve:

- Phase 1: Deploy Sidecar for logging and analytics.
- Phase 2: Integrate Interceptor for high-risk queries or regulated data.
- Phase 3: Replace Sidecar with Interceptor for all queries, using the Sidecar only for debugging.

INTEGRATION PATTERNS

HOW THE INTERCEPTOR POSTURE INTEGRATES WITH APPLICATIONS

The Interceptor is designed to be non-intrusive, requiring minimal changes to existing applications. It supports three integration patterns:

1. SDK INJECTOR

- Description: A lightweight SDK is injected into the application code. The SDK intercepts AI queries and routes them through the Interceptor.
- Pros: Full control over query routing; supports complex logic (e.g., retry on failure).
- Cons: Requires code changes.
- Best For: AI-native startups, custom applications.

2. EDGE DEPLOYMENT

- Description: The Interceptor is deployed as a sidecar container or service mesh (e.g., Istio, Linkerd) alongside the application. Queries are automatically routed through the Interceptor.
- Pros: No code changes; transparent to the application.
- Cons: Requires infrastructure support (e.g., Kubernetes, service mesh).
- Best For: Cloud-native enterprises, microservices architectures.

3. ZERO-CODE CONFIGURATION

- Description: The Interceptor is deployed as a reverse proxy or API gateway. Applications send queries to the Interceptor's endpoint, which forwards compliant queries to the AI model.
- Pros: No code or infrastructure changes.
- Cons: All queries must route through the Interceptor; may introduce a single point of failure.
- Best For: Legacy enterprises, quick start deployments.

INTEGRATION PATTERN COMPARISON

Integration Pattern | Latency Overhead | Complexity | Best For

SDK Injector | Low | High | AI-Native Startups

Edge Deployment | Medium | Medium | Cloud-Native Enterprises

Zero-Code Configuration | High | Low | Legacy Enterprises

CONCLUSION: COMPLIANCE AS INFRASTRUCTURE

THE SHIFT FROM DASHBOARDS TO INFRASTRUCTURE

Traditional compliance tools focus on dashboards: visual interfaces for auditing logs and detecting violations after the fact. This approach is reactive, manual, and prone to gaps. The Interceptor posture treats compliance as infrastructure: a foundational layer that prevents violations before they occur.

Paradigm | Compliance as Dashboard | Compliance as Infrastructure

Enforcement | Passive (after execution) | Active (before execution)

Evidence | Logs | Cryptographic artifacts

Latency | Low | Minimal overhead

Failure Mode | Violation logged, damage done | Query blocked, no compliance gap

Integration | Manual audits | Automated, real-time

Scalability | Limited by human auditors | Limited by hardware

WHY THIS MATTERS FOR CTOS, VPS, AND CISOS

- Regulatory Assurance: Active Enforcement provides mathematical proof of compliance, reducing legal and financial risk.

- Operational Efficiency: Automated enforcement eliminates manual audits, freeing up engineering and compliance teams.

- Security: Cryptographic artifacts prevent tampering and provide non-repudiation.

- Future-Proofing: As AI regulations evolve, Active Enforcement ensures organizations can adapt quickly by updating policies, not rewriting applications.

THE NEXT FRONTIER

The next frontier in AI compliance is network-layer enforcement: embedding compliance checks into the network itself (e.g., at the API gateway or load balancer). This would enable universal compliance, zero-trust AI, and global policies across all AI systems, on-premises and in the cloud.

The Interceptor posture is the first step toward this vision, proving that compliance can be active, real-time, and tamper-proof-not just a checkbox, but a guarantee.

RTFCT Technologies, 2026