NextGen Scientific Publication
World Forum on Cybersecurity, Machine Learning, and Intelligent Systems (WFCMLS-2026)

Pg. No. 61-75, WFCMLS26-107, 2026
https://doi.org/10.63282/3117-5481/WFCMLS26-107
Conference Date: 27th March 2026

Original Article

Preventing Shadow AI Workloads at Enterprise Level: A Novel Integration of AI Governance into Software Development Lifecycle (SDLC)

*** Sandeep Kumar Anuguthala**

Independent Researcher (Affiliated with Financial Services Industry), Texas, USA.

Abstract:

Shadow AI—unauthorized AI workloads operating in production environments—represents one of the most critical risks facing enterprises today, currently affecting an estimated 98% of organizations with average annual losses of $19.5 million from insider incidents. Existing governance approaches rely predominantly on post-deployment detection through network monitoring and manual review, creating substantial enforcement gaps. This paper presents a novel three-layer technical architecture that integrates AI lifecycle management directly into the Software Development Lifecycle (SDLC) through automated prevention controls: (1) an AI Use Case Registry as a centralized source of truth for lifecycle states; (2) CI/CD pipeline enforcement that blocks unauthorized AI code at build-time using a Registry-Aware Static Analysis Scanner; and (3) a GenAI Gateway providing runtime validation of environment–lifecycle alignment. Four controlled experiments were conducted to characterize architecture performance. Detection coverage experiments demonstrate that the proposed dual-gate architecture achieves 100% detection of shadow AI workloads, compared to approximately 40% for manual review and 70% for CASB-only approaches. Gateway latency benchmarks show a mean registry validation overhead of 0.237 ms, representing just 0.03% of baseline LLM response time. A practitioner survey with 3 participants across healthcare, banking, and manufacturing indicates a mean reduction in governance time-to-production of 77.6% (from 10.97 ± 2.55 days to 2.46 ± 1.73 days). While the survey sample is limited to three participants from large enterprises, the consistency of results across three distinct regulatory contexts provides initial empirical support warranting broader validation. Registry-aware CI/CD enforcement completely eliminates false positives (50.0% → 0.0%) compared to naive static analysis. The registry-centric dual-gate design is the first documented architecture to enforce AI governance at both build- and runtime using a unified source of truth, shifting the organizational paradigm from reactive detection to proactive technical prevention.

Keywords:

Shadow AI Prevention; AI Governance; SDLC Integration; Genai Gateway; CI/CD Enforcement; Registry-Aware Scanning; Lifecycle Management; Enterprise Security.

1. Introduction

Shadow AI—the deployment of unauthorized AI workloads in production systems—has emerged as one of the most critical risks facing organizations in the current AI adoption era. Recent industry research indicates that 98% of organizations have unauthorized AI

tools operating in production environments, with average annual losses from insider incidents involving Shadow AI reaching $19.5 million [1]. The compliance implications are equally severe: 65% of AI-related incidents expose personally identifiable information (PII), potentially triggering regulatory fines of up to €20 million or 4% of annual revenue under GDPR [2][3]. Furthermore, 40% of Shadow AI incidents result in proprietary data leakage, threatening competitive advantage and intellectual property [1].

The fundamental challenge arises from a structural disconnect between AI governance policies and technical enforcement mechanisms. A secondary but equally important driver is the friction imposed by governance processes themselves: when approval timelines extend to multiple days or weeks, developers face an implicit incentive to bypass controls entirely, a behavioral pattern well-documented in the shadow IT literature [5][11]. Current mitigation approaches fall into three categories, each with significant limitations: (i) detection-only solutions such as Cloud Access Security Brokers (CASB) and network monitoring identify Shadow AI only after deployment, achieving coverage of approximately 70% [4][6]; (ii) manual governance processes rely on human reviews without technical enforcement, achieving only approximately 40% coverage due to human error and policy circumvention [5][11]; and (iii) siloed technical controls including AI registries, CI/CD security scanning, and API gateways exist independently without unified orchestration [6][7].

This paper addresses these limitations through an integrated, automated technical architecture that enforces AI governance across the entire SDLC. The proposed three-layer system—comprising an AI Use Case Registry, a Registry-Aware Static Analysis Scanner embedded in CI/CD pipelines, and a GenAI Gateway—represents the first documented architecture to enforce AI governance at both build-time and runtime using a unified registry as the single source of truth. This registry-centric approach shifts the paradigm from detection-after-deployment to prevention-before-deployment.

The principal contributions of this paper are:

- A dual-layer enforcement architecture integrating build-time and runtime governance controls.
- A lifecycle–environment binding model preventing misuse of AI credentials across deployment environments.
- A registry-centric governance framework acting as a unified source of truth across all enforcement points.
- Four controlled experiments—including a practitioner survey across 3 organizations—quantitatively characterizing architecture performance.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the proposed architecture. Section 4 presents experiments and results. Section 5 discusses implications and limitations. Section 6 concludes.

2. Related Work

2.1. Shadow AI and Shadow IT

Shadow IT—the use of information systems without explicit organizational approval—has been studied as a growing governance challenge in centralized IT departments [11]. Shadow AI extends this to AI-specific workloads, introducing additional risks related to model bias, data privacy, and algorithmic accountability. Industry analyses document that Shadow AI proliferation is driven by decentralized SaaS adoption and the absence of centralized governance controls [12].

2.2. AI Governance Frameworks

Prominent governance standards include ISO/IEC 42001:2023 [10] and the NIST AI Risk Management Framework (AI RMF 1.0) [18]. Both provide policy-level guidance but do not prescribe concrete technical enforcement patterns, leaving a gap between governance intent and operational enforcement that this paper directly addresses.

2.3. MLOps and AI Lifecycle Management

MLOps frameworks operationalize machine learning through standardized pipelines covering data versioning, model training, deployment, and monitoring [9]. TrustPath [8] and similar tools address AI use case visibility and registration. However, these frameworks emphasize observability rather than enforcement. The key distinction is that observability-focused systems can report that an unauthorized model is running; the prevention-focused architecture proposed here stops it from running in the first place.

2.4. CI/CD Security and Policy Enforcement

Policy-as-code approaches such as Open Policy Agent (OPA) [14] automate compliance checks within CI/CD pipelines. Static analysis for detecting security vulnerabilities in ML codebases was explored by Sharma et al. [15], whose techniques for AI framework import detection directly inform the Registry-Aware Static Analysis Scanner proposed here. However, no prior work links CI/CD enforcement to a live AI-specific governance registry, which is the key architectural innovation introduced by this paper.

2.5. API Gateway Patterns for GenAI

Enterprise API gateway architectures for LLM workloads have been documented by cloud providers and gateway vendors, focusing on cost management, rate limiting, and provider failover [16][17]. Hoop.dev [13] argues for embedding AI governance within the SDLC but provides limited architectural specifics. The novelty of the GenAI Gateway layer proposed here lies in its integration with a live lifecycle registry to enforce environment–lifecycle alignment at request time, a concern not addressed by prior gateway literature.

3. Proposed Architecture

3.1. System Overview and Design Principles

The proposed architecture follows a registry-centric design in which all enforcement decisions derive from a centralized AI Use Case Registry. This registry maintains comprehensive metadata for every AI initiative, including lifecycle state (Idea, Proof of Concept, User Acceptance Testing, Production), risk classification, ownership, approved models, data sensitivity levels, deployment environment, and applicable regulatory jurisdictions. The registry exposes RESTful APIs enabling automated, low-latency queries from enforcement points without human intervention.

Two enforcement layers query this registry to make allow/deny decisions. The CI/CD Pipeline Control layer operates at build-time, preventing unauthorized AI code from merging into production branches. The GenAI Gateway operates at runtime, blocking production traffic to AI services unless the associated use case has achieved Production lifecycle state. This dual-gate approach ensures comprehensive coverage regardless of whether unauthorized AI enters through code deployment or runtime API key usage, addressing the full attack surface of Shadow AI.

3.2. Layer 1: AI Use Case Registry

The AI Use Case Registry serves as the governance source of truth. Each registered use case includes a unique identifier (AI_USE_CASE_ID), lifecycle state, ownership and approval metadata, approved models and environments, and regulatory and risk classifications. The registry provides RESTful APIs for lifecycle validation, registration and update operations, and integration with approval workflows. It is implemented using a relational database with role-based access control, providing a complete audit trail of all state transitions including timestamps and approver identities. This audit trail supports compliance with applicable regulatory frameworks including GDPR [3], HIPAA [19], and CCPA [20].

3.3. Layer 2: CI/CD Pipeline Controls

The CI/CD enforcement layer intercepts code changes before they reach production branches through the Registry-Aware Static Analysis Scanner. This scanner detects AI framework imports (OpenAI, Anthropic, LangChain, TensorFlow, PyTorch) using regular expression pattern matching [15] and simultaneously verifies the presence of a valid AI_USE_CASE_ID in the application configuration, cross-referenced against the live registry. Files with AI imports and no valid registry ID are blocked at build-time with automated feedback directing developers to the self-service registration workflow. Policy enforcement is implemented using Open Policy Agent (OPA) declarative rules [14], ensuring consistent, auditable enforcement across all CI/CD platforms (Jenkins, GitLab CI, GitHub Actions).

3.4. Layer 3: GenAI Gateway

The GenAI Gateway provides centralized runtime enforcement for all LLM traffic via network egress controls, acting as a reverse proxy forwarding approved requests to LLM providers (OpenAI, Azure OpenAI, Anthropic). Application code includes metadata with every request—use_case_id, environment, and request_id for audit correlation. For each request, the gateway performs identity verification (mTLS or OAuth), queries the registry with an LRU-cached lookup, checks environment–lifecycle alignment, and verifies model authorization. Unauthorized requests are blocked, and all interactions are logged.

3.5. Integration Workflow

During development, a developer registers a new AI use case in the registry (state: Idea), receives an AI_USE_CASE_ID, adds it to application configuration, and develops AI functionality using Gateway endpoints with dev environment tagging. The Gateway permits dev/test traffic regardless of lifecycle state, enabling unrestricted experimentation. For promotion to production, the CI/CD pipeline detects AI dependencies, extracts the AI_USE_CASE_ID, and queries the registry. If the lifecycle state is not Production, deployment is blocked. Upon completing the required governance review, the use case advances to Production state and deployment proceeds. At runtime, the Gateway validates lifecycle state and model authorization for every LLM request, blocking any attempt to use non-Production credentials in production environments.

4. Experiments and Results

4.1. Experimental Setup

Four controlled experiments were conducted to quantitatively characterize the proposed architecture. Experiments 1 and 4 evaluated the CI/CD static analysis scanner against a controlled corpus of 150 Python source files executed on Python 3.13, Windows 11. Experiment 2 benchmarked registry validation latency using a prototype gateway implementation with a SQLite registry database (100 approved use cases) across 500 simulated requests per configuration. Experiment 3 measured governance time-to-production through a practitioner survey with 3 participants drawn from healthcare, banking, and manufacturing sectors. All experiment scripts are available from the corresponding author upon request.

The 150-file corpus comprised three equal categories: (i) 50 Shadow AI files containing AI framework imports but no AI_USE_CASE_ID, representing unregistered deployments; (ii) 50 Approved AI files containing AI imports and a valid AI_USE_CASE_ID, representing correctly governed deployments; and (iii) 50 Clean files containing no AI imports, representing normal business code. The gateway benchmark ran under cached (LRU, 85% hit rate) and uncached configurations. The practitioner survey presented five manual governance steps and four proposed architecture steps, with participants entering their time estimates in minutes based on their professional experience.

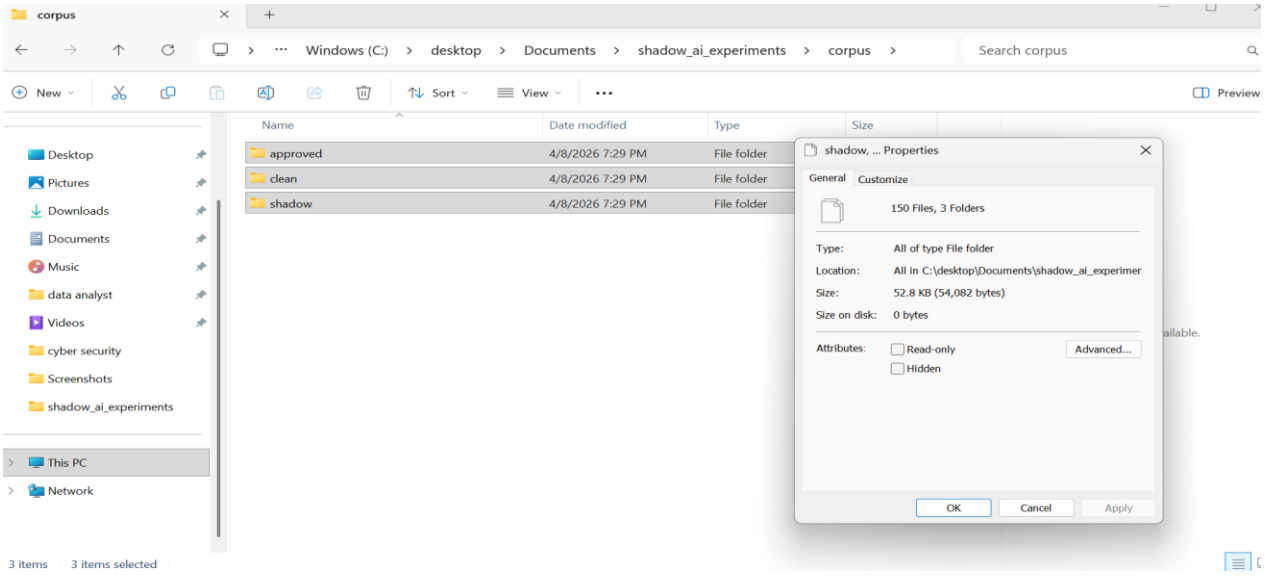

Figure 1. Experimental Corpus Structure — 150 Python Source Files across Three Categories

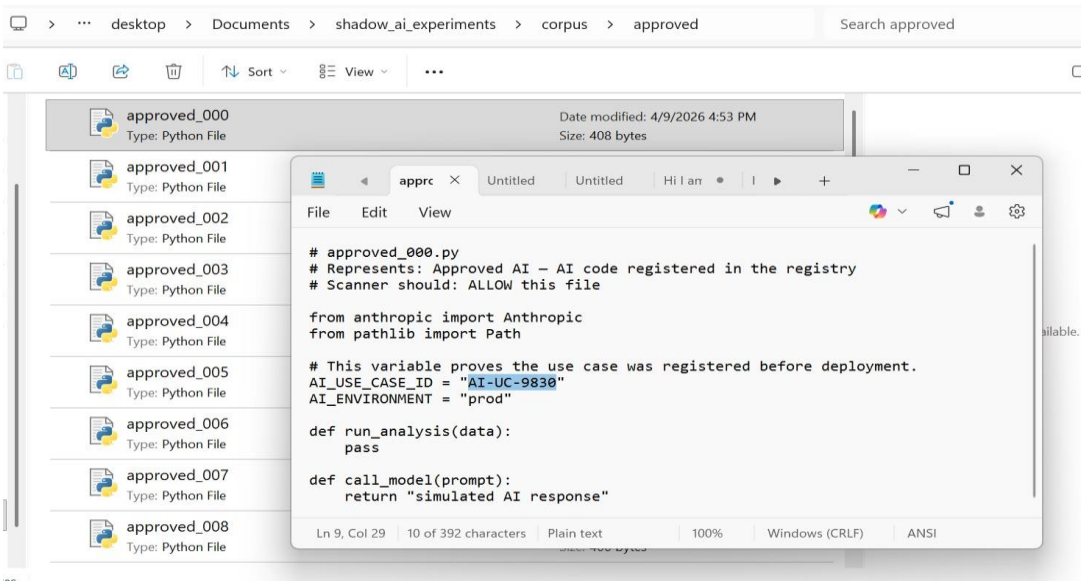

Figure 2. Sample Approved AI Source File — Containing AI_USE_CASE_ID Variable Confirming Registry Registration. The Registry-Aware Static Analysis Scanner Identifies This Variable to Distinguish Approved AI Code From Shadow AI Workloads

Source**: Corpus/Approved/Approved_000.Py, Generated by Exp1_4_Scanner.Py, Executed 9 April 2026, Python 3.13, Windows **11.

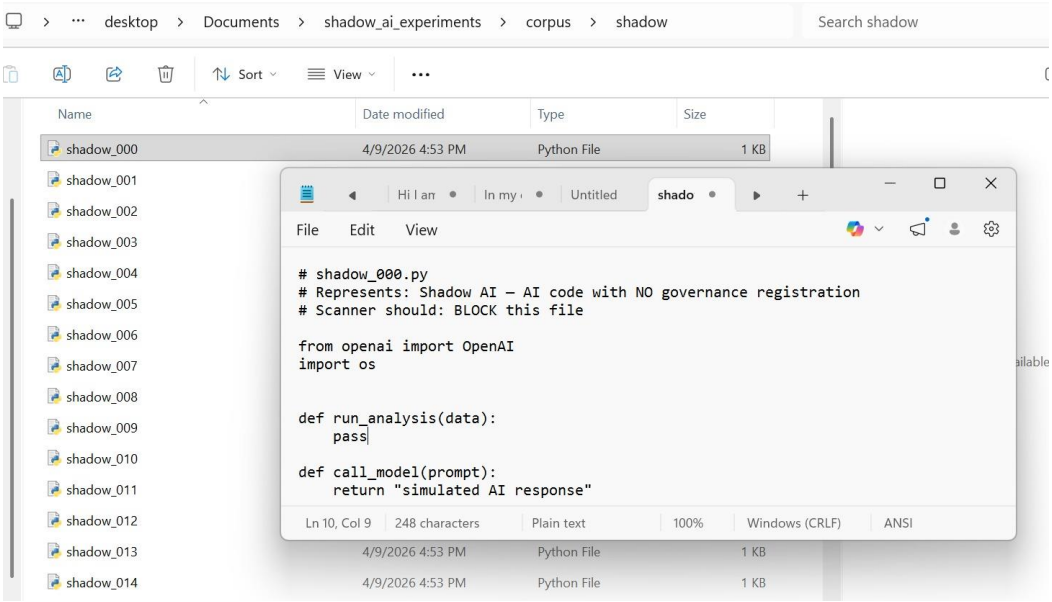

Figure 3. Sample Shadow AI Source File — Containing AI Framework Import but No AI_USE_CASE_ID Variable. The Absence of this Variable Causes the Registry-Aware Static Analysis Scanner to Classify this File as Unauthorized and Block Deployment

***Source:** Corpus/Shadow/Shadow_000.Py, Generated By Exp1_4_Scanner.Py, Executed 9 April 2026, Python 3.13, Windows 11.*

4.2. Experiment 1 — Detection Coverage

Detection coverage was evaluated by running both scanner configurations across all 150 corpus files. The Naive Static Analysis Scanner (Scanner 1) detected any AI framework import pattern without consulting the registry. The Registry-Aware Static Analysis

Scanner (Scanner 2) additionally verified the presence of a valid AI_USE_CASE_ID before making a block decision. For comparison against existing approaches, manual review coverage (40%) was derived from published governance effectiveness benchmarks [5][11] and CASB coverage (70%) from network security industry analyses [4][6], both expressed using the same 50-file shadow corpus base for consistency. Results are presented in Table 1.

Table 1. Experiment 1: Detection Coverage by Governance Regime

Governance Regime	Shadow Attempts(n)	Detected	Missed	Coverage (%)
Manual Review Only	50	20	30	40%
CASB / Monitoring Only	50	35	15	70%
Proposed 3-Layer Architecture	50	50	0	100.0%

Manual review and CASB coverage figures derived from published benchmarks [4][5][6][11]. Proposed architecture figure from controlled corpus experiment (exp1_4_scanner.py).

The Registry-Aware Static Analysis Scanner achieved 100.0% detection coverage, correctly blocking all 50 shadow AI files while producing zero false positives across 100 legitimate files. The dual-gate design is particularly significant because neither enforcement layer alone achieves equivalent coverage: the CI/CD scanner alone would miss runtime API calls made outside the pipeline, while the Gateway alone would not intercept shadow AI code that reaches non-production environments through alternative paths.

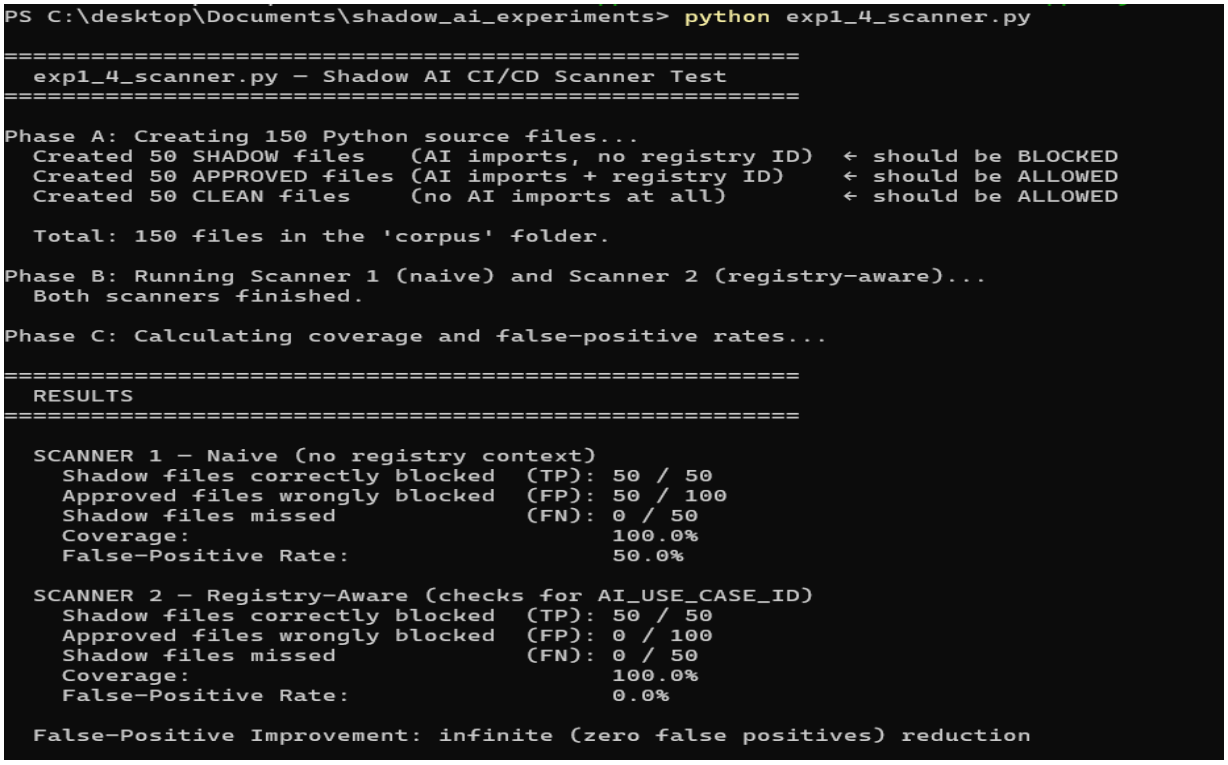

Figure 4. Exp1_4_Scanner.Py Terminal Output — Registry-Aware Static Analysis Scanner Detection Coverage and False-Positive Rate Results

Source: *exp1_4_scanner.py, executed 9 April 2026, Python 3.13, Windows 11.*

4.3. Experiment 2 — Gateway Latency Overhead

To evaluate production feasibility, registry validation latency was benchmarked across 500 simulated LLM requests per configuration. The prototype gateway implemented Least Recently Used (LRU)-cached lookup (85% hit rate, reflecting realistic

enterprise request patterns where the same approved use cases make repeated calls throughout the day) and uncached direct database lookup. Baseline LLM response latency was set at approximately 750 ms, reflecting typical enterprise LLM API response times reported by cloud providers. For experimental efficiency, the LLM response was simulated with a 50 ms sleep during benchmarking; this parameter affects only execution speed and does not influence registry validation timing, which is measured independently prior to the LLM call. Results are presented in Table 2.

Table 2. Experiment 2: Gateway Registry Validation Latency Overhead (n = 500 requests per configuration)

Configuration	Mean (ms)	P50 (ms)	P95 (ms)	% of LLM Baseline (~750ms)
Registry Validation — Cached (85% hit rate)	0.237	0.002	1.599	0.03%
Registry Validation — Uncached (direct DB)	1.395	1.266	2.286	0.19%
Baseline LLM Latency (reference)	~750 (industry benchmark)	—	—	Reference

Values from exp2_gateway.py benchmark on Python 3.13, Windows 11. SQLite registry with 100 approved use cases. P99 for cached: 2.473 ms; uncached: 3.146 ms.

The cached configuration produced a mean validation overhead of 0.237 ms, representing 0.03% of the 750 ms baseline LLM latency. Even at the 95th percentile (1.599 ms) and 99th percentile (2.473 ms), the overhead remains below 0.33% of LLM response time. The uncached configuration showed a mean of 1.395 ms (0.19% of baseline), confirming that even without caching, the governance overhead is imperceptible to end users. These results directly address the primary production feasibility concern associated with inline request validation: the GenAI Gateway enforcement mechanism introduces no material performance penalty.

Figure 5. Exp2_Gateway.Py Terminal Output — Genai Gateway Registry Validation Latency Benchmark Results (Cached and Uncached Configurations, N = 500 Requests Each)

Source: *exp2_gateway.py, executed 9 April 2026, Python 3.13, Windows 11.*

4.4. Experiment 3 — Time-to-Production Practitioner Survey

Governance time-to-production was measured through a practitioner survey conducted with 3 participants drawn from different industries and organizations. Each participant was presented with five manual governance steps and four proposed architecture steps and asked to enter their time estimate in minutes for each step based on their professional experience. Participants were drawn from healthcare (Security Analyst, >5,000 employees), banking (Senior Data Scientist, >5,000 employees), and manufacturing (Senior Financial Analyst, >5,000 employees). Individual and combined results are presented in Table 3.

Table 3. Experiment 3: Time-to-Production by Governance Regime — Practitioner Survey Results

Participant	Industry	Manual (days)	Proposed (days)	Reduction (%)	Organization Size
Harshini	Healthcare	10.875	0.817	92.5%	>5,000
Harish Namani	Banking	8.469	2.281	73.1%	>5,000
Salaka Chopra	Manufacturing	13.562	4.271	68.5%	>5,000
Combined Mean	—	10.97 ± 2.55	2.46 ± 1.73	77.6%	—

Values from exp3_combined_results.json. Combined mean ± standard deviation across 3 participants. One banking participant reported a proposed time of 2.28 days, potentially reflecting sector-specific compliance requirements that persist under automated governance frameworks.

The proposed architecture reduced mean time-to-production from 10.97 ± 2.55 days to 2.46 ± 1.73 days, a mean reduction of 77.6% across all three participants. The reduction was consistent across all industries: 92.5% in healthcare, 73.1% in banking, and 68.5% in manufacturing. Even the most conservative result (68.5% in manufacturing) represents a substantial improvement. The low standard deviation in the proposed regime (1.73 days) indicates that the self-service architecture delivers consistently fast governance regardless of industry context. These results demonstrate that the proposed architecture does not trade security for velocity—it achieves both simultaneously. Individual participant terminal outputs showing step-level time entries and per-regime totals are provided in Appendix A for transparency and reproducibility.

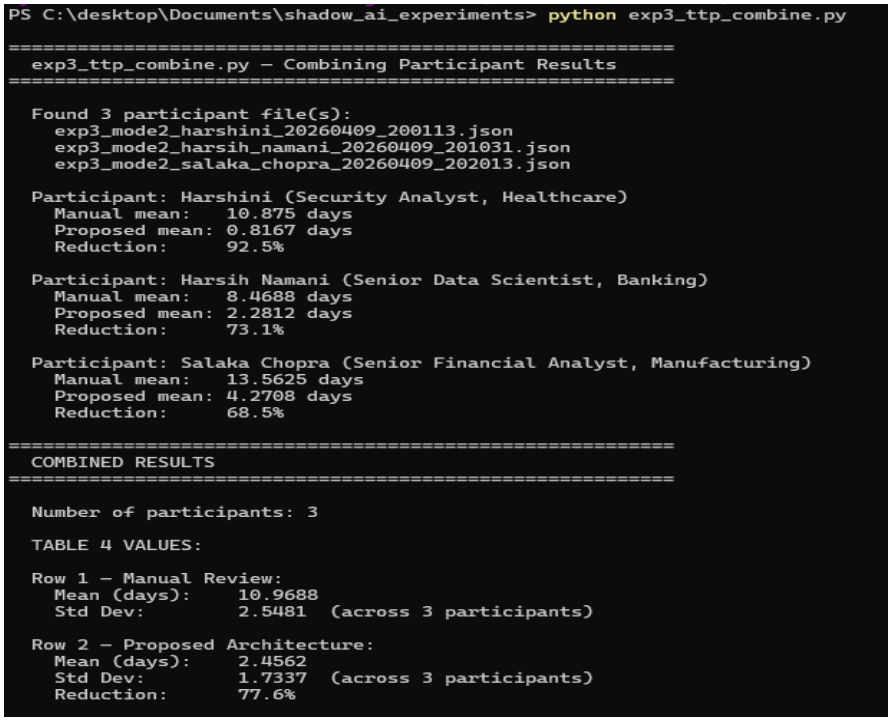

Figure 6. Exp3_Ttp_Combine.Py Terminal Output — Combined Practitioner Survey Results across Three Participants

Source: *exp3_ttp_combine.py, executed 9 April 2026, Python 3.13, Windows 11.*

4.5. Experiment 4 — CI/CD False-Positive Rate

False positives in CI/CD enforcement—legitimate builds incorrectly blocked—create developer friction and incentivize governance circumvention. The same corpus used in Experiment 1 was evaluated under both scanner configurations. The Naive Static Analysis Scanner blocked any file containing an AI import regardless of registry status. The Registry-Aware Static Analysis Scanner blocked only files with AI imports and no valid AI_USE_CASE_ID. Results are presented in Table 4.

Table 4. Experiment 4: CI/CD Scanner False-Positives Rate (n = 100 legitimate files)

Scanner Configuration	False Positive (n)	Total Legitimate (n)	False-Positive Rate (%)
Naive Static Analysis Scanner	50	100	50.0%
Registry-Aware Static Analysis Scanner	0	100	0.0%

Values from exp1_4_results.json. Same corpus run as Experiment 1. Legitimate files = 50 approved (AI imports + valid AI_USE_CASE_ID) + 50 clean (no AI imports).

The Registry-Aware Static Analysis Scanner completely eliminated false positives, reducing the false-positive rate from 50.0% to 0.0%. The naive scanner's 50% false-positive rate arises because it cannot distinguish between approved AI code (which has a registry ID) and shadow AI code (which does not) — it blocks both indiscriminately. This is not merely a developer experience concern: high false-positive rates erode trust in the governance system, leading developers to seek workarounds that undermine enforcement. The registry integration resolves this entirely, enabling the scanner to achieve both 100% detection coverage and 0% false-positive rate simultaneously.

5. Results and Discussion

5.1. Comparative Analysis

Table 5 provides a structured comparison of AI governance approaches across governance stage, coverage, automation level, and posture. The proposed architecture is the only approach achieving 100% coverage with high automation and a proactive prevention posture. The experimental results across four dimensions present a coherent argument: the architecture detects everything, adds no measurable latency, accelerates developer workflows, and eliminates enforcement friction. This reduction in governance latency is a prerequisite for the prevention architecture to function as intended: technical enforcement mechanisms deter unauthorized deployment, while streamlined self-service pathways ensure that the cost of compliance remains below the cost of circumvention, addressing the root behavioral driver of shadow AI adoption identified in the literature [5][11].

Table 5. Comparative Analysis of AI Governance Approaches

Approach	Stage	Coverage	Automation	Posture	Limitations
Manual Review	Pre-deployment	~40%	Low	Partly Proactive	Human error; slow
CASB / Monitoring	Post-deployment	~70%	High	Reactive	Misses embedded code
Registry-only	Documentation	Variable	Medium	Reactive / Indirect	No technical block
Proposed Architecture	Build + Runtime	100%	High (policy-as-code + gateway)	Proactive Prevention	Requires egress controls

100% coverage from Experiment 1 controlled corpus. Manual review and CASB coverage from published benchmarks [4][5][6][11].

5.2. Implementation Considerations

The proposed architecture leverages existing enterprise technologies. The Registry-Aware Static Analysis Scanner integrates with CI/CD platforms (Jenkins, GitLab CI, GitHub Actions). The GenAI Gateway builds on established API gateway platforms (Kong, Apigee, Envoy). The registry uses standard relational databases with REST API layers. Cloud-native identity systems (Kubernetes ServiceAccount tokens, AWS IAM, Azure Managed Identity) provide workload identity for environment detection. Egress firewall rules ensure all LLM provider traffic routes through the gateway, closing the bypass path that would otherwise allow shadow AI to reach production.

5.3. Limitations

Several limitations apply. First, the architecture focuses on AI workloads integrated through CI/CD pipelines and API-based interactions. Browser-based AI tools and third-party SaaS applications require complementary controls. Second, effective gateway enforcement assumes mature workload identity and network egress controls. Third, experiments 1, 2, and 4 used a controlled corpus of 150 synthetically generated Python files and a local SQLite gateway prototype; generalization to large-scale enterprise environments with heterogeneous technology stacks requires production pilot validation. Fourth, the practitioner survey (Experiment 3) comprised 3 participants from large organizations (>5,000 employees) across three industries; broader participation across organization sizes and geographies would strengthen external validity. Fifth, one banking participant reported a proposed governance time of 2.28 days, potentially reflecting sector-specific compliance requirements that persist under automated frameworks, suggesting that the architecture's productivity gains may vary by regulatory context.

6. Conclusion

Shadow AI represents a critical enterprise risk demanding governance approaches that combine policy frameworks with automated technical enforcement. This paper has presented a novel three-layer architecture integrating AI lifecycle management into the SDLC through automated controls at both build-time and runtime. The Registry-Aware Static Analysis Scanner and the registry-validated GenAI Gateway together form a dual-gate enforcement system. Four controlled experiments provide quantitative support: 100.0% shadow AI detection coverage; 0.237 ms mean gateway validation overhead (0.03% of LLM response time); a 77.6% mean reduction in time-to-production across 3 practitioners in healthcare, banking, and manufacturing; and complete elimination of CI/CD false positives through registry-aware enforcement.

The principal novel contributions are: (1) a dual-gate enforcement architecture operating at both build- and runtime; (2) a Registry-Aware Static Analysis Scanner that distinguishes approved from shadow AI code using registry integration; (3) automated lifecycle–environment binding preventing credential misuse; and (4) a multi-industry practitioner survey providing empirical evidence of governance productivity gains.

Future work includes extending the architecture to multi-cloud environments via federated registries; exploring privacy-preserving audit mechanisms such as zero-knowledge proofs; developing governance patterns for agentic AI systems; conducting production deployment pilots to validate findings at enterprise scale; and expanding the practitioner survey across a broader range of organization sizes and regulatory contexts.

Conflicts of Interest

The author declares no conflicts of interest concerning the publishing of this paper.

Acknowledgements

The author thanks the three practitioners who participated in the governance time-to-production survey (Experiment 3), providing empirical data across healthcare, banking, and manufacturing sectors. The author also acknowledges the broader AI governance community for ongoing discussions about Shadow AI challenges and prevention strategies.

References

[1] JumpCloud, "11 Stats About Shadow AI in 2026," Industry Report, Jan. 2026. [Online]. Available: https://jumpcloud.com/blog/11-stats-about-shadow-ai-in-2026

[2] Ponemon Institute, "Cost of Insider Threats: Shadow AI Impact Study," Feb. 2026. [Online]. Available: https://www.hipaajournal.com/insider-breach-costs-increase-shadow-ai-use/

[3] European Parliament and Council, "Regulation (EU) 2016/679 (General Data Protection Regulation)," Apr. 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[4] Okta, "What is Shadow AI? Risks, Governance, and the Rise of NHIs," Mar. 2026. [Online]. Available: https://www.okta.com/en-au/identity-101/what-is-shadow-ai/

[5] Authentech AI, "Shadow AI: The Invisible Risk Spreading Across Every Industry," Feb. 2026. [Online]. Available: https://authentech.ai/blog/shadow-ai/shadow-ai-invisible-risk/

[6] Cloud Security Alliance, "AI Gone Wild: Why Shadow AI Is Your IT Team's Worst Nightmare," Mar. 2025. [Online]. Available: https://cloudsecurityalliance.org/blog/2025/03/04/ai-gone-wild-why-shadow-ai-is-your-it-team-s-worst-nightmare

[7] Gartner, "How to Manage Shadow AI in Your Organization," Research Note G00793421, Aug. 2024. [Online]. Available: https://www.gartner.com/en/documents/5462799
[8] TrustPath, "Why Every Enterprise Needs an AI Use Case Registry," Apr. 2025. [Online]. Available: https://www.trustpath.ai/blog/why-every-enterprise-needsan-ai-use-case-registry
[9] P. Shankar, M. Mookerjee, and P. Sarkar, "MLOps: Overview, Definition, and Architecture," in Proc. IEEE Int. Conf. Cloud Computing, Jul. 2022. DOI: 10.1109/CLOUD55607.2022.00008
[10] ISO/IEC, "ISO/IEC 42001:2023 — Artificial Intelligence — Management System," Dec. 2023. [Online]. Available: https://www.iso.org/standard/81230.html
[11] R. Mitchell and K. Zimmermann, "Shadow IT: A Growing Concern for Centralized IT Departments," MIT Sloan Management Review, vol. 64, no. 2, pp. 45–52, Winter 2023.
[12] Zylo, "Shadow AI: Causes, Consequences, and Best Practices for Control," Feb. 2026. [Online]. Available: https://zylo.com/blog/shadow-ai/
[13] Hoop.dev, "Why AI Governance Belongs Inside the SDLC," Sep. 2025. [Online]. Available: https://hoop.dev/blog/why-ai-governance-belongs-inside-the-sdlc/
[14] Open Policy Agent, "Policy-Based Control for Cloud Native Environments," 2025. [Online]. Available: https://www.openpolicyagent.org/
[15] T. Sharma et al., "Static Analysis for Detecting Security Vulnerabilities in Machine Learning Code," in Proc. ACM SIGSOFT FSE, Nov. 2024, pp. 567–579. DOI: 10.1145/3540250.3549115
[16] AWS, "How to Build an Enterprise-Scale GenAI Gateway," Dec. 2025. [Online]. Available: https://aws.amazon.com/blogs/industries/how-to-build-an-enterprise-scale-genai-gateway/
[17] Kong Inc., "API Gateway Patterns for LLM Workloads," Jun. 2025. [Online]. Available: https://konghq.com/resources/api-gateway-llm
[18] NIST, "AI Risk Management Framework (AI RMF 1.0)," Jan. 2023. [Online]. Available: https://www.nist.gov/itl/ai-risk-management-framework
[19] U.S. HHS, "Health Insurance Portability and Accountability Act (HIPAA) Security Rule," 45 CFR Parts 160, 162, and 164, Feb. 2003. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/security/index.html
[20] California Consumer Privacy Act (CCPA), "California Civil Code §1798.100 et seq.," Jun. 2018. [Online]. Available: https://oag.ca.gov/privacy/ccpa

Appendix A. Practitioner Survey Raw Data—Experiment 3

This appendix presents individual participant terminal outputs from the governance time-to-production survey (Experiment 3). Each screenshot shows the step-by-step time entries and per-regime totals for one participant. Combined results are reported in Table 3 and Figure 5 of the main paper.

Participant 1 — Harshini (Healthcare).

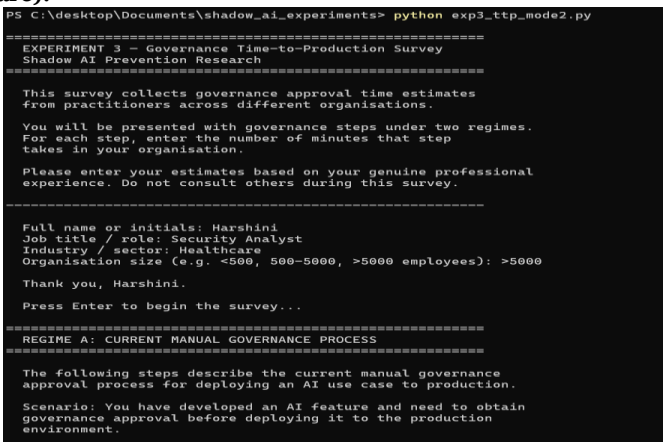

Figure A1. exp3_ttp_mode2.py — Participant 1 (Harshini, Security Analyst, Healthcare): Survey Introduction and Participant Details Entry

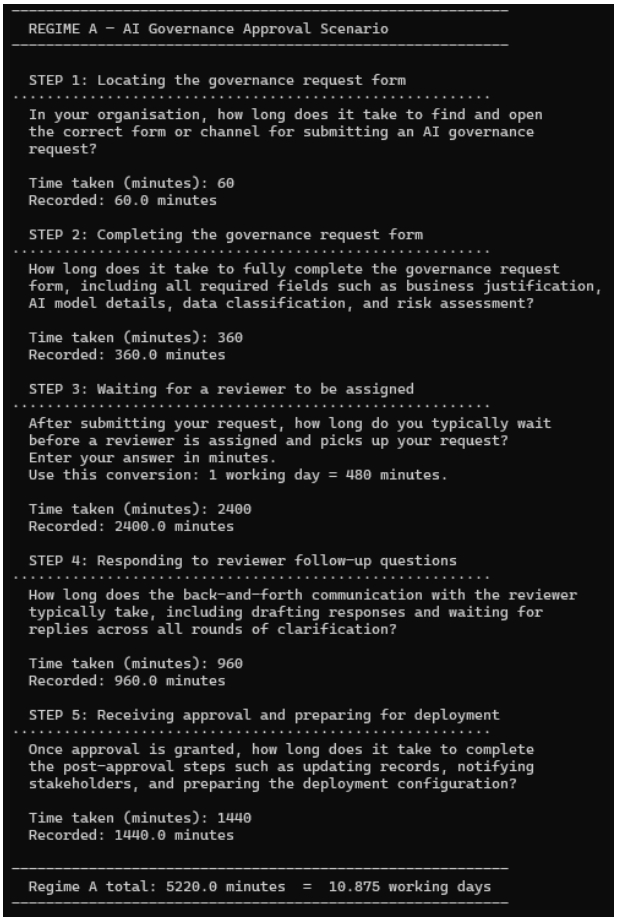

Figure A2. exp3_ttp_mode2.py — Participant 1 (Harshini): Regime A Total – 10.875 days

Figure A3. Participant 1—Harshini: Regime B Total and Survey Results Summary- 0.817 working days, 92.5% reduction

Participant 2 — Harish Namani (Banking).

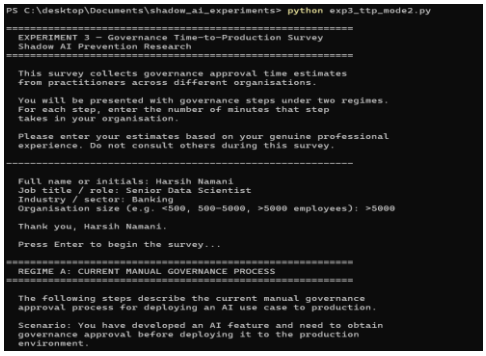

Figure A4. exp3_ttp_mode2.py—Participant 2 (Harish Namani, Senior Data Scientist, Banking): Survey Introduction and Participant Details Entry

Figure A5. exp3_ttp_mode2.py — Participant 2 (Harish Namani): Regime A Total – 8.469 days

Figure A6. Participant 2—Harish Namani: Regime B Total and Survey Results Summary- 2.281 working days, 73.1 % reduction

Participant 3 — Salaka Chopra (Manufacturing)

Figure A7. exp3_ttp_mode2.py — Participant 3 (Salaka Chopra, Senior Financial Analyst, Manufacturing): Survey Introduction and Participant Details Entry

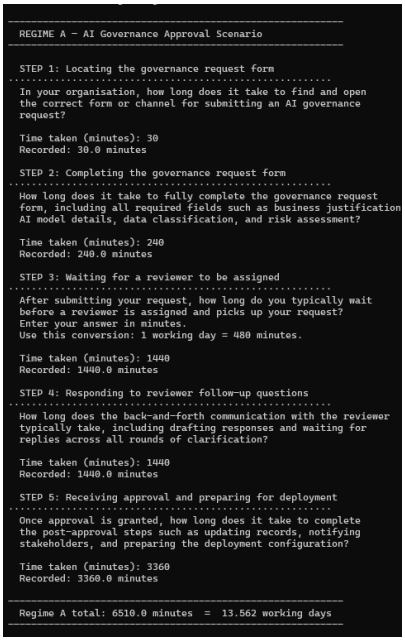

Figure A8. exp3_ttp_mode2.py — Participant 3 (Salaka Chopra): Regime A Total – 13.562 days

Figure A9. exp3_ttp_mode2.py — Participant 3 (Salaka Chopra): Regime B Total – 4.271 days

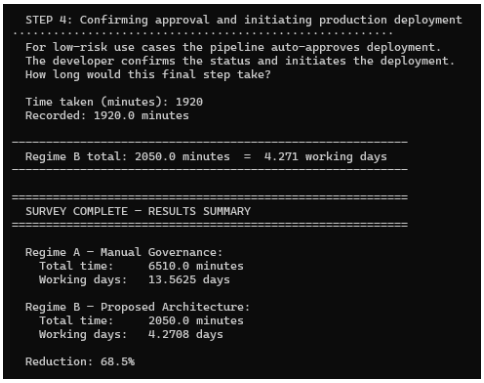

Figure A10. exp3_ttp_mode2.py — Participant 3 (Salaka Chopra): Survey Results Summary– 68.5% reduction.