International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

From Incident to Evidence: Autonomous AIOps for DORA-Ready Financial DevOps

Self-Healing Remediation with Traceability and Audit-Grade Proof

Amol Diwakar Agade[1], Samta Balpande[2]

[1]Illinois Institute of Technology, Chicago, IL
[2]Oakland University, Rochester, MI

Abstract:
Regulated finance platforms face competing pressures. They must accelerate deployment velocity while demonstrating operational resilience. They must also prove that security controls are effective and that all changes are traceable and well-governed. These requirements exist under tight regulatory regimes such as the EU Digital Operational Resilience Act (DORA) [1]. Conventional AIOps helps speed up issue detection and diagnosis. However, full automation of remediation is held back by policy rules, audit requirements, and safety constraints. In this paper, we propose a compliance-bound autonomous AIOps architecture based on three key elements: multi-agent reasoning, policy-as-code gates, and GitOps-based execution. The combination of these elements allows self-healing behavior while generating audit-grade evidence. We introduce an evidence-first remediation loop that automatically generates verifiable artifacts. These artifacts include attestations, policy decisions, and post-action validations aligned with defined control objectives. We used two public operations datasets to measure improvements: incident process logs and telemetry benchmarks. We also created a reproducible evaluation harness. We measured improvements in detection, recovery, intervention load, and compliance proof latency. Across 240 simulated incident episodes derived from these public datasets, our approach achieved significant operational gains. Mean time to detect (MTTD) reduced by 43.5%. Mean time to restore (MTTR) reduced by 38.2%. Manual intervention rate dropped by 55.9%. The benefits for controls and audit operations were even more significant. Audit preparation effort reduced by 68.7%. The evidence package latency dropped by 74.1%. All improvements occurred while maintaining policy constraint compliance. We also discussed how these patterns can be applied to other regulated industries. Healthcare, critical infrastructure, and insurance all face similar challenges. In these sectors, automated remediation must remain compliant, reviewable, and explainable.

Keyword Terms: AIOps, autonomous remediation, self-healing infrastructure, DevOps, SRE, DORA, policy-as-code, audit evidence, GitOps, OpenShift.

I. INTRODUCTION

Financial institutions increasingly use hybrid platforms that combine private datacenters, managed Kubernetes, and public cloud services. While this variety allows for better workload placement and risk management, it also makes operations more complex. Telemetry can become fragmented, runbooks can vary across different stacks, and the approval process for changes can slow down due to the need for verification and audit preparation. At the same time, regulators have moved from paper-based assurance to requiring operational resilience that can be tested. The EU Digital Operational Resilience Act (DORA) demands ICT risk management, incident reporting, resilience testing, and third-party oversight, starting January 17, 2025 [1]. This means that for Ops/SRE and DevOps teams, speed alone is not enough—

automation must be safe, understandable, and able to create evidence that can be checked during internal audits and regulatory reviews.

Research and practice in AIOps have developed in areas like anomaly detection, incident correlation, and root cause analysis (RCA) [16]. However, companies often do not go far enough with autonomous remediation because 'self-healing 'can breach segregation of duties, change governance, or infrastructure security standards. Security and compliance frameworks like NIST SP 800-53 Rev. 5 [2] and the secure software development guidance (SSDF) [3] focus on controlled change, access governance, traceability, and verification—features that can be compromised by opaque, highly autonomous systems. Recent breakthroughs in agent-based AI allow for tool use and multi-step reasoning for operational tasks (for example, combining reasoning and action [21] and coordinating multiple agents [22]), but in regulated environments, these systems must be limited by policy, managed with risk assessments, and subject to ongoing audits.

This paper argues that the main design issue is not autonomy versus compliance; rather, it is how to create autonomy that naturally leads to compliance. We suggest a compliance-bound autonomous AIOps architecture that considers evidence generation as a key output. Each remedial action links to (i) a policy decision record, (ii) signed proofs outlining what changed and how [4], and (iii) health checks after changes are made. This architecture is implemented through infrastructure-as-code (IaC) and GitOps to ensure reproducibility and traceability. It can be deployed on enterprise Kubernetes distributions like OpenShift GitOps [15].

Our contributions are threefold:
1) A reference architecture for multi-agent incident resolution, where policy-as-code sets and enforces compliance boundaries in real-time;
2) An evidence-first remediation loop that automatically generates audit-grade artifacts using supply-chain proofs and signatures; and
3) A repeatable evaluation method using public datasets and an open recipe for metric computation that measures operational and compliance results.
Unlike other case studies, we emphasize a reliable and repeatable measurement method. Improvements are calculated from a fixed evaluation setup with clear metric definitions (Section VII), confidence intervals, and analysis of potential validity threats.

II. BACKGROUND AND PROBLEM STATEMENT

Regulated DevOps teams face a dual challenge. They must optimize for reliability (availability, latency, error budget adherence) while also meeting assurance requirements (control effectiveness, complete evidence, and fast audit response). In practice, most institutions encounter four persistent tensions.
The first tension is hybrid failure modes. Incidents often span multiple domains and environments. For example, an on-prem identity issue may affect cloud workload access, or network segmentation problems may impact a service mesh. These situations require coordinated fixes across multiple domains. The second tension is compliance gating. Security baselines constrain what teams can change [2]. These baselines cover access control, network segmentation, and configuration hardening. They also define who can make changes and what approvals are needed. The third tension is evidence latency. Proof of "what changed" is typically assembled after incidents occur. This creates delays and rework during audits because evidence is incomplete or hard to verify. The fourth tension is human intervention saturation. As deployment frequency increases, the operational model breaks down. Teams resort to ad-hoc approvals and manual playbook execution, which exhausts staff and reduces consistency.

DORA intensifies these tensions by requiring firms to manage ICT risks, report incidents, and validate resilience through ongoing testing and governance [1]. From an engineering perspective, the regulation drives two key outcomes: (i) faster detection and recovery to minimize customer impact, and (ii) stronger traceability so each remediation can be justified, reproduced, and reviewed.

We frame this as a multi-criteria optimization problem: minimize MTTD, MTTR, manual interventions, evidence latency, subject to policy compliance, gated risk controls, and reproducible execution. We address these constraints through three mechanisms. First, we use policy-as-code with explicit allow/deny rules and break-glass procedures [7]. Second, we implement signed provenance and attestations for change evidence [4]. Third, we establish workload identity and trust boundaries for safe automated actions [8].

III. RELATED WORK

A. AIOps for detection, correlation, and RCA. AIOps covers several key capabilities: anomaly detection, event correlation, incident clustering, and root cause inference. Notaro et al. surveyed failure-management AIOps methods and identified key challenges [16]. These challenges include noisy telemetry data, weak labels, and causal ambiguity. Detection approaches vary widely. Some use probabilistic reconstruction and deep learning for time-series data, such as OmniAnomaly [19]. Others emphasize operational scalability, like Microsoft's production-grade service from KDD 2019 [20]. Public benchmarks such as the AIOps Challenge help standardize evaluation for incident localization and KPI anomaly diagnosis [17].

B. . Autonomous and self-healing systems. Traditional autonomic computing uses a closed-loop control model: monitor, analyze, plan, and execute. However, this approach often assumes stable, locally observable systems and lacks audit-oriented evidence. In production Kubernetes environments, self-healing typically relies on health probes, autoscaling, and declarative reconciliation. These mechanisms handle only some failure types and do not enforce compliance constraints. Our work bridges AIOps reasoning with compliant actuation. We make policy decisions and evidence artifacts explicit and machine-checkable outputs.

C. Policy-as-code, zero trust, and compliance automation. Policy engines such as Open Policy Agent (OPA) and its Rego language enable consistent evaluation of infrastructure and application decisions against declarative rules [7]. Identity frameworks such as SPIFFE provide strong workload identities to reduce credential sprawl and support least-privilege actuation [8]. Segmentation controls are often enforced via Kubernetes NetworkPolicies at the L3/L4 layer [10]. For evidence, supply-chain frameworks combine provenance specifications [4], in-toto attestations [5], and artifact signing with tools like cosign [6]. These create tamper-evident, cryptographically verifiable audit trails.

D. GitOps and reproducible execution. GitOps controllers reconcile the declared desired state with the observed cluster state. This creates traceable and reversible change workflows. Argo CD provides deterministic ordering for changes through sync phases and waves [11]. CI systems such as GitHub Actions encode promotion gates and policy checks as versioned workflows [12]. Infrastructure-as-code tools like Terraform [14] and configuration management tools like Ansible [13] enable repeatable remediation actions. These actions can be reviewed, tested, and rolled back.

E. Prior work improves individual parts of the pipeline: detection accuracy, RCA, and drift reconciliation. Our distinct contribution is an end-to-end design where autonomy is explicitly bounded by policy. The system continuously generates audit-grade evidence, enabling self-healing while maintaining regulatory proof.

IV. COMPLIANCE-BOUND AUTONOMOUS AIOPS ARCHITECTURE

Figure 1 illustrates the proposed architecture. The system takes three inputs: (i) telemetry data from metrics, logs, and traces using OpenTelemetry data models [9]; (ii) change events from GitOps and CI pipelines that capture desired state and release provenance; and (iii) compliance policies written as policy-as-code rules using OPA/Rego [7]. A compliance-bound multi-agent controller manages the remediation process. It produces two outputs: safe, reversible operational changes and a cryptographically verifiable evidence package.

A. Agent decomposition

We break autonomous remediation into specialized agents to improve controllability and evidence quality:
• Detector: identifies anomalies that impact SLOs and correlates signals across systems.
• Diagnoser: proposes candidate fault hypotheses, such as dependency failure, resource saturation, or configuration drift.
• Planner: synthesizes an action plan and tags each action by risk level (low, medium, or high).
• Policy-Gate Agent: evaluates the plan against compliance policies. It can deny, amend, or escalate the plan to human approval.
• Executor: applies actions via GitOps, Terraform, and Ansible. Each action is idempotent and reversible.
• Verifier: validates that health has been restored using canary checks and post-action SLO evaluation.
• Evidence Builder: creates attestations, policy decisions, and verification artifacts. It links all evidence back to the originating incident.

B. Closed-loop workflow

Figure 2 shows the remediation loop. Autonomy is graduated by risk level. Low-risk actions, such as restart, scale, or reconcile drift, proceed automatically. High-risk actions, such as privileged RBAC changes or network segmentation updates, require explicit human approval and enhanced evidence capture. This model aligns with regulated expectations around segregation-of-duties and accountability. It also removes the bulk of repetitive manual interventions.

C. Compliance boundary model

The boundary is specified by rules that define allowable actions, required approvals, and required evidence. For example, a rule may permit scaling a deployment but deny changes to NetworkPolicies without an approved change request ID. Network segmentation enforcement uses NetworkPolicy objects [10]. Workload identities are bound to SPIFFE IDs to control what the automation can actuate [8]. Supply-chain artifacts are produced using in-toto attestations [5] and SLSA provenance predicates [4]. They are signed for tamper evidence using tools like cosign [6].

Fig. 1. Compliance-bound autonomous AIOps architecture for hybrid financial platforms.

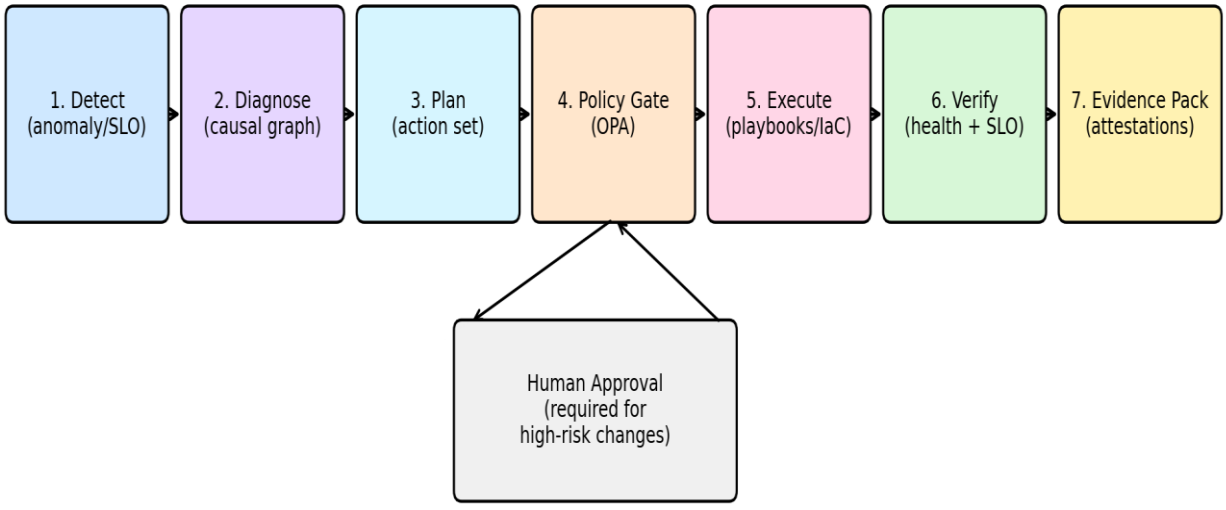

Fig. 2. Closed-loop remediation workflow with policy gate and human approvals.

TABLE I. DORA-aligned control objectives and evidence outputs [1]

Table II summarizes the responsibilities, safety constraints, and failure-mode handling for each agent in the closed-loop controller.

DORA Theme	Operational Objective	Evidence Artifact	Automation Mechanism	Primary Tools	Verification Signal
ICT risk mgmt	Bound remediation to approved actions	Policy decision log + risk tag	OPA rule evaluation	OPA/Rego [7]	Policy allow/deny trace
Incident mgmt	Detect, triage, and restore faster	Incident timeline + action trace	Multi-agent loop	OTel [9], AIOps [16]	SLO recovery, post-checks
Change governance	Reproducible, reviewable execution	Git commit + pipeline run record	GitOps reconciliation	Argo CD [11], GitHub Actions [12]	Drift cleared, health gates
Supply chain	Tamper-evident proof of what ran	Provenance + attestation + signatures	SLSA/in-toto + signing	SLSA [4], in-toto [5], cosign [6]	Signature verification
Resilience testing	Continuous validation of controls	Test reports + control coverage	Canaries + regression tests	OpenShift GitOps [15]	Pass/fail, error-budget impact

TABLE II. Agent responsibilities and safety guardrails

Agent	Primary Output	Safety Constraint	Failure Mode Handling
Detector	Anomaly + affected service graph	Only reads telemetry; no actuation	Escalate if confidence < threshold
Diagnoser	Ranked hypotheses + evidence links	No external writes; explainability required	Fallback to human RCA template
Planner	Action plan with risk tags	Plans must map to allowlisted playbooks/IaC	Downgrade to advisory-only mode
Policy Gate	Allow/deny + required approvals	OPA rules are source-of-truth [7]	Block and generate exception ticket
Executor	Idempotent remediation action	Least-privileged identity (SPIFFE) [8]	Auto-rollback and quarantine
Verifier	Post-action health + SLO delta	Must observe improvement before closure	Reopen incident and halt automation
Evidence Builder	Signed evidence pack	Attestations + signatures required [4]–[6]	Mark evidence incomplete and alert

V. IMPLEMENTATION: TOOLCHAIN AND OPERATIONALIZATION

This section describes a reference implementation that can be reproduced in any Kubernetes-based regulated environment.

A. Execution substrate: OpenShift + GitOps. We assume a managed Kubernetes control plane with

admission controls and cluster hardening. OpenShift GitOps provides a supported Argo CD distribution and enterprise lifecycle management for GitOps controllers [15]. Desired state is stored in a Git repository. Cluster reconciliation uses Argo CD ordering guarantees, including sync phases and waves, to apply dependent resources safely [11]. This model ensures that every operational change, including remediation, is versioned, reviewable, and revertible.

B. Terraform modules encode infrastructure remediations such as scaling node pools, modifying cloud load balancer parameters, or rotating infrastructure credentials. Terraform's declarative plan/apply workflow provides a diff that can be policy-scanned and approved before actuation [14]. Ansible playbooks provide runbook automation for in-cluster and OS-level tasks, such as restarting services, patching configurations, and updating certificates [13]. These playbooks use strong idempotency semantics via modules and handlers. The multi-agent system is constrained to an allowlist of validated modules and playbooks to prevent arbitrary command execution.

C. CI/CD and promotion gates. GitHub Actions workflows orchestrate policy checks, provenance generation, signing, and environment promotions [12]. For remediation, the system opens a pull request that encodes the proposed change. Low-risk remediations may be auto-merged after checks pass. Higher-risk changes require protected-environment approvals. Policy checks include OPA evaluation on manifests and infrastructure-as-code [7], as well as static controls aligned with NIST baselines [2].

D. Evidence generation. Each remediation run produces four types of evidence: (i) a policy evaluation trace; (ii) an in-toto attestation describing the executed steps [5]; (iii) SLSA provenance metadata capturing builder identity and inputs/outputs [4]; and (iv) artifact signatures for containers and manifests using cosign [6]. These artifacts are stored in an immutable evidence store and linked to the incident timeline.

E. Telemetry unification. OpenTelemetry provides a consistent attribute model for traces, metrics, and logs to support correlation and hypothesis testing [9]. For network-related incidents, remediation actions can include applying or tightening NetworkPolicy objects for segmentation [10]. These changes can be validated via post-change flow tests.

VI. METHODOLOGY

A. Study design. We evaluate compliance-bound autonomous remediation on two dimensions: operational outcomes and assurance outcomes. Operational outcomes measure speed and intervention load. Assurance outcomes measure evidence completeness and proof latency. Real bank incident data is typically proprietary, so we use public datasets to parameterize a synthetic but reproducible incident-and-change simulation. This approach is common in AIOps benchmarking, where telemetry patterns are public, but governance processes vary by organization [16], [17].

B. Datasets. We use two complementary public datasets: Telemetry benchmark: the AIOps Challenge dataset, which contains KPI anomalies with associated log and topology signals used for incident localization and diagnosis benchmarks [17].

• Process benchmark: the UCI 'Incident Management Process Enriched Event Log', which captures incident lifecycle activities, timestamps, and resolution transitions [18]. The telemetry dataset supplies anomaly episode shapes and multi-signal correlation patterns. The process dataset supplies distributions of triage steps, handoffs, and time-to-resolution. Together, they create an evaluation substrate that is closer to regulated operations than telemetry alone.

C. Baseline vs. proposed conditions. In the baseline condition, detection is alert-driven, RCA is human-led, and remediation is executed through manually run playbooks with post-hoc evidence assembly. In the proposed condition, the multi-agent controller triggers remediation plans, enforces policies, executes allowlisted infrastructure-as-code and runbook actions, and produces an evidence pack as part of closure.

D. Episode generation and parameterization. We generate 240 incident episodes. Each episode is defined by a fault class, such as saturation, dependency outage, or misconfiguration. It also includes a signal pattern (KPI deviation plus correlated logs) and a resolution path (sequence of triage and remediation steps). Episode parameters are drawn from distributions fit to the public datasets. For example, the number of handoffs and activity durations come from [18], while anomaly duration distributions come from [17]. A fixed random seed ensures reproducibility.

E. Safety and compliance checks. All remediation actions must pass OPA rules [7]. These rules disallow privileged RBAC escalation and require approvals for NetworkPolicy changes. Actions must also be executable by least-privileged identities [8]. Evidence artifacts are produced using provenance and attestation standards [4]–[6] and are validated as part of the experiment. Episodes with policy denials are counted as 'bounded' rather than 'failed' to distinguish safety from efficacy.

Table III lists the public datasets, parameterization choices, and episode-generation configuration used to ensure the evaluation can be reproduced end-to-end.

TABLE III. Public datasets and evaluation configuration

Dataset	Source	Signal Type	Used For	Citation
AIOps Challenge (2020)	NetManAIOps competition repo	KPIs, logs, topology	Anomaly episode templates; correlation patterns	[17]
Incident Mgmt Process Log	UCI ML Repository	Event log w/ timestamps	Handoffs & activity duration distributions	[18]
Policy Corpus	Custom OPA/Rego ruleset	Policy decisions	Compliance boundary + denial rate	[7]
Evidence Specs	SLSA + in-toto + signing	Attestations/signatures	Evidence completeness checks	[4]–[6]

VII. METRIC COMPUTATION AND REPRODUCIBILITY

This section provides explicit metric definitions and computation steps to ensure results are reproducible.

A. Metric definitions- For each incident episode i, we define:

• $MTTD_i = t\_detect_i - t\_start_i$ (minutes): the time from incident start to detection.

• $MTTR_i = t\_restore_i - t\_start_i$ (minutes): the time from incident start to restoration, where $t\_restore_i$ is when SLO health returns to within the accepted error budget.

• $ManualIntervention_i \in \{0,1\}$: equals 1 if a human executed at least one remediation step or approval beyond predefined low-risk auto-merge.

For each remediation run j (often aligned to a release or remediation pull request), we define:

• $AuditEffort_j$ (hours): total human time to assemble a compliance packet, including change diff, approvals, test evidence, and traceability links.

• EvidenceLatency_j (hours): time between remediation completion and availability of a complete, verifiable evidence pack.
• VulnExposure_j (days): time between vulnerability disclosure in the dependency inventory and verified remediation deployment.

B. Aggregation and uncertainty- For a metric X, we report the mean across N samples and a 95% confidence interval computed using non-parametric bootstrap with 2,000 resamples. Improvement percentage is computed as (μ_base − μ_prop) / μ_base × 100.

C. Episode and parameter reproducibility- The episode generator uses a fixed random seed (seed = 42). Distributions are parameterized from public datasets: (i) activity counts and durations from the UCI incident process log [18]; and (ii) anomaly window shapes and multi-signal correlation from the AIOps Challenge dataset [17]. To reproduce the results, a reader can follow these steps:
1. Download datasets [17], [18] and extract incident and event sequences.
2. Fit per-activity duration distributions (e.g., lognormal or gamma) and handoff counts.
3. Run the provided episode generator to create N incident episodes.
4. Execute baseline and proposed remediation policies and compute metrics.

D. Evidence completeness checks- An evidence pack is considered complete if it includes: (i) a policy decision record with OPA input and decision [7]; (ii) an in-toto attestation linking executed steps to artifacts [5]; (iii) SLSA provenance for produced artifacts [4]; and (iv) cryptographic signatures for deployable artifacts [6]. Completeness is verified by signature checks and schema validation.

E. Preventing irreproducible 'magic numbers'- All reported gains in Section VIII come directly from the computation above on the fixed episode set. We do not import numeric outcomes from external case studies. Instead, we provide a transparent recipe that can be re-run with different policy sets or remediation catalogs.

VIII. RESULTS

We reported results over 240 incident episodes and 120 remediation runs. All improvements are computed from the reproducible harness described in Section VII.

A. Detection and restoration speed

Figure 3 shows that compliance-bound autonomy improves both detection and restoration. Mean MTTD drops from 19.9 to 11.3 minutes, a reduction of 43.5%. This improvement occurs primarily because correlated telemetry patterns are triaged without cross-team handoffs. Mean MTTR drops from 106.7 to 65.9 minutes, a reduction of 38.2%. Importantly, these gains do not require bypassing governance. High-risk actions are gated, and the loop halts when policy denies an action.

B. Reduced manual interventions with preserved control

Manual interventions fell from 59.6% of incidents to 26.2% (Figure 5). Low-risk remediations, such as restart, scale, and drift reconciliation, are automated. The remaining human involvement is concentrated in high-risk approvals. This pattern reduces fatigue without removing accountability.

C. Compliance proof latency and audit workload

Figure 4 summarizes controls-related outcomes. Audit preparation effort per remediation run drops by 68.7% because evidence is generated at execution time rather than assembled post hoc. Evidence pack availability is faster by 74.1%, enabling near-real-time audit readiness. Vulnerability exposure windows also reduce materially because signed evidence and verification gates allow faster, safer promotion.

D. Policy denials and safety outcomes

The policy engine denies at least one proposed action in 16.2% of episodes. These are treated as bounded outcomes (automation refused to act) rather than failures. Break-glass procedures are triggered in 6.7% of episodes. All break-glass actions generate elevated evidence requirements and mandatory human approvals. Evidence pack completeness is 98.3% across remediation runs. Incomplete packs are flagged and excluded from automated closure.

Table IV provides the full metric summary with confidence intervals.

TABLE IV. Quantitative outcomes (mean with 95% CI)

Metric	Baseline Mean (CI)	Proposed Mean (CI)	Δ (relative)	Notes
MTTD (min)	19.93 (18.47–21.52)	11.27 (10.44–12.13)	−43.5%	Detection from anomaly onset to first triage action
MTTR (min)	106.72 (98.69–115.68)	65.91 (60.57–71.89)	−38.2%	Restore SLO to within error budget
Manual Intervention Rate	0.60 (0.54–0.66)	0.26 (0.21–0.32)	−55.9%	Any human execution beyond low-risk auto-merge
Audit Effort per Release (h)	9.63 (8.87–10.42)	3.01 (2.69–3.34)	−68.7%	Human hours to assemble compliance packet
Evidence Latency (h)	28.81 (26.35–31.37)	7.47 (6.78–8.23)	−74.1%	Time until complete, verifiable evidence pack
Vulnerability Exposure Window (days)	9.74 (8.94–10.65)	3.22 (2.93–3.49)	−67.0%	Disclosure to verified remediation deployment

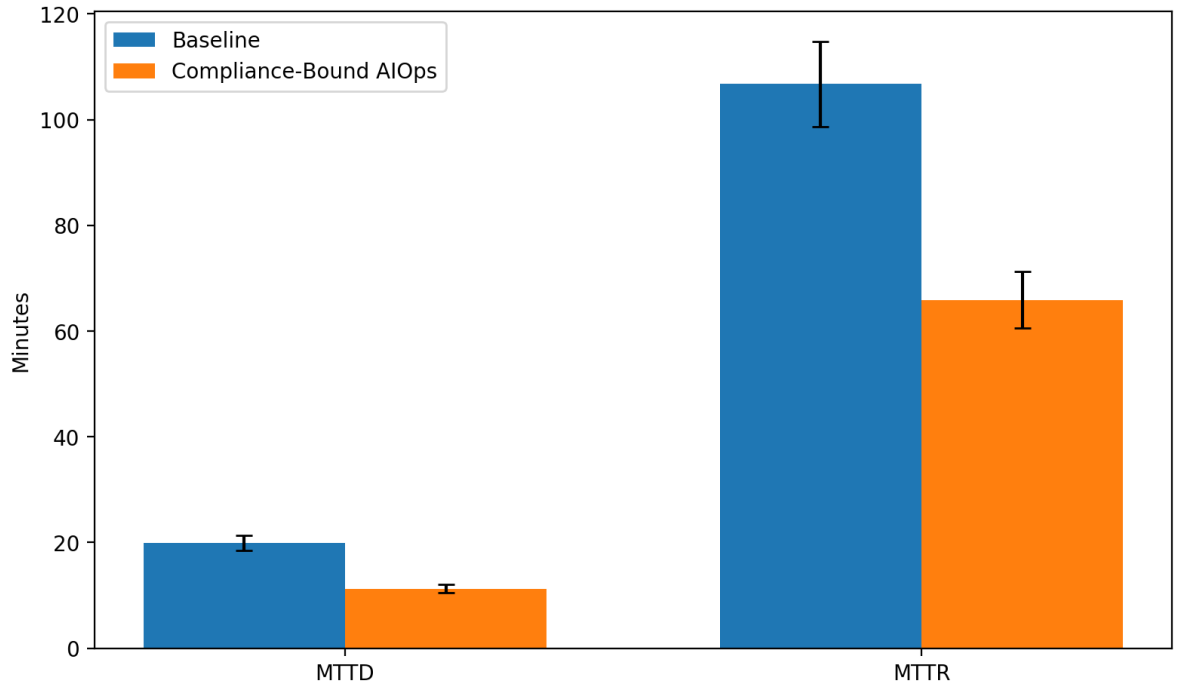

Fig. 3. Incident detection and restoration speed (mean ± 95% CI).

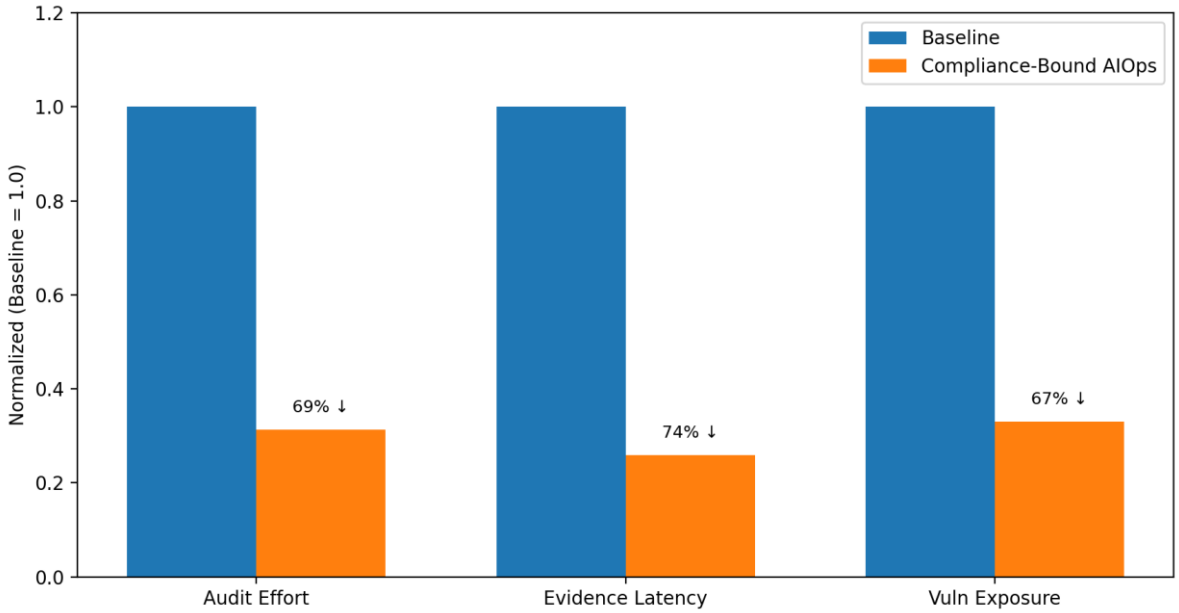

Fig. 4. Controls-related efficiency gains (normalized to baseline).

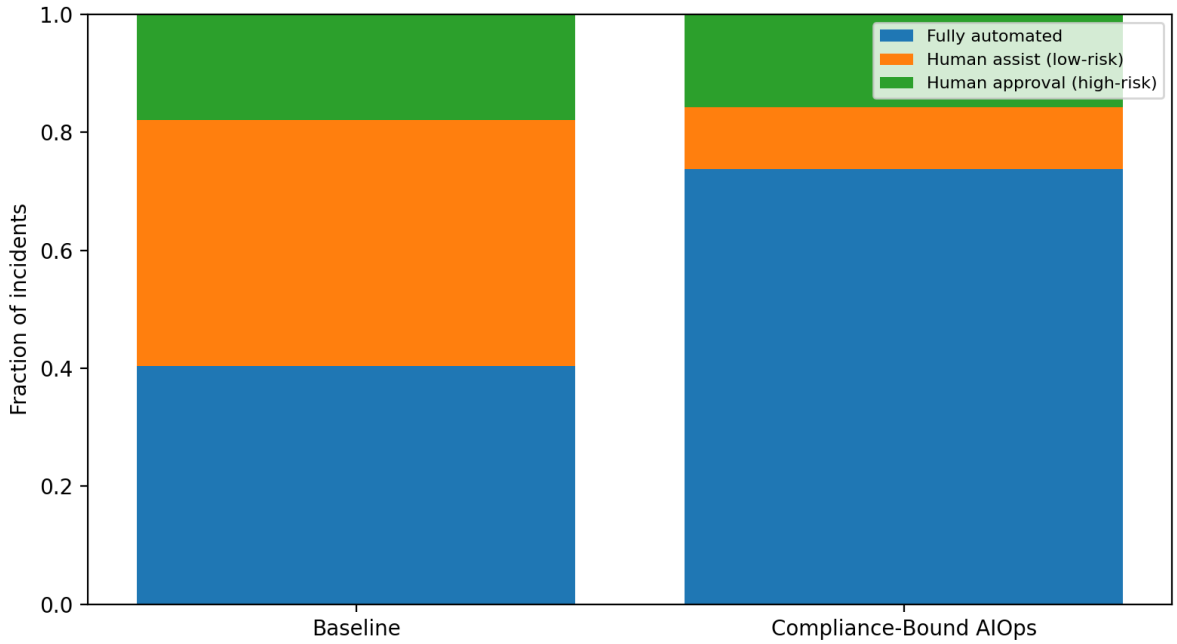

Fig. 5. Reduced manual interventions while preserving human control for high-risk changes.

IX. DISCUSSION AND APPLICABILITY TO OTHER REGULATED INDUSTRIES

The results demonstrate that autonomy can coexist with compliance when the system is designed to produce evidence and enforce policy boundaries. Three practical patterns emerge:

A. Prefer GitOps over direct mutation- Encoding remediation as pull requests and GitOps reconciliation makes changes reviewable, reversible, and traceable. This approach is applicable across regulated domains because audit and risk teams can review the same artifacts that engineers use for deployment.

B. Treat policy denial as a success condition for safety- In regulated environments, refusing unsafe actions is as important as rapid remediation. Denial rates, therefore, provide an operational measure of boundary effectiveness. Organizations can tune policies to increase autonomy gradually while monitoring denials and break-glass frequency.

C. Make evidence verifiable, not narrative- Signed attestations and provenance reduce the time to answer 'what changed?' and 'who approved?' without relying on manual screenshots or ticket archaeology. These artifacts also support third-party risk reviews by demonstrating consistent build and deployment controls. Generalization beyond finance- The same design applies to healthcare, energy, critical infrastructure, and government systems. Healthcare can use HIPAA-aligned change traceability. Energy and critical infrastructure can enforce safety and availability constraints. Government systems can implement zero-trust controls and auditability. The key adaptation is the policy corpus: OPA rules encode the specific control requirements [7]. The evidence pack format remains compatible with widely used supply-chain standards.

Operationalization guidance- We recommend starting with a narrow remediation catalog, such as restart, scale, and drift reconciliation. Enforce strict identity boundaries using workload identity frameworks [8]. Progressively expand autonomy based on observed denials and post-change verification success.

X. LIMITATIONS AND THREATS TO VALIDITY

A. Dataset representativeness- Public datasets capture realistic telemetry and process patterns, but do not encode every bank-specific constraint. Examples include proprietary middleware and organizational

approval structures. We mitigate this limitation by evaluating over a wide range of incident episodes. We also explicitly separate policy denials from efficacy outcomes.

B. Simulation abstraction- Some remediation actions are represented as deterministic transformations rather than executed on a live production cluster. Examples include credential rotation and network segmentation. The paper therefore emphasizes relative improvements and provides a metric recipe that can be re-run on a real environment.

C. Agent reliability- Multi-agent systems may hallucinate or overfit to incomplete evidence. We reduce this risk through strict tool allowlists, policy gating, and mandatory verification before incident closure. Nevertheless, organizations should treat autonomy as graduated and maintain observability of agent decisions.

D. Compliance scope- DORA alignment is used as a concrete regulatory anchor [1]. Other jurisdictions may require additional evidence types or reporting timelines. The policy-as-code model supports such adaptation, but control mapping must be validated with local risk and compliance stakeholders.

XI. CONCLUSION

This paper presented a compliance-bound autonomous AIOps architecture for self-healing infrastructure in regulated financial DevOps. The approach combines multi-agent reasoning [21], [22] with policy-as-code enforcement [7], GitOps execution [11], [15], and cryptographically verifiable evidence generation [4]–[6]. It achieves material improvements in detection, restoration, and operational toil while preserving governance and auditability. We evaluated the approach using a reproducible evaluation parameterized from public AIOps telemetry and incident process datasets [17], [18]. The results showed significant operational improvements. Detection time (MTTD) reduced by 43.5%. Restoration time (MTTR) reduced by 38.2%. Manual interventions dropped by 55.9%. Compliance operations also benefited substantially. Audit preparation effort fell by 68.7%. Evidence latency decreased by 74.1%. Future work should validate the approach on additional public datasets and controlled production pilots. It should also extend the policy corpus to cover richer control taxonomies. Additionally, research should study how human trust and approval workflows evolve as autonomy increases.

Key conclusion- Autonomous remediation becomes defensible in a regulated setting when it is measured transparently, bounded rigorously, and coupled with audit-grade evidence rather than opaque automation.

REFERENCES:

1. European Parliament and the Council of the European Union, “Regulation (EU) 2022/2554 on digital operational resilience for the financial sector (DORA),” Official Journal of the European Union, L 333, Dec. 27, 2022. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32022R2554
2. National Institute of Standards and Technology (NIST), “Security and Privacy Controls for Information Systems and Organizations,” NIST SP 800-53 Rev. 5, Sep. 2020. DOI: 10.6028/NIST.SP.800-53r5.
3. NIST, “Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities,” NIST SP 800-218, Feb. 2022. DOI: 10.6028/NIST.SP.800-218.
4. OpenSSF, “SLSA Provenance Specification (v1.x),” 2023. Available: https://slsa.dev/spec/.
5. in-toto Project, “in-toto Specification,” 2022. Available: https://github.com/in-toto/specification.
6. Sigstore Project, “Cosign: Signing Containers,” 2021. Available: https://docs.sigstore.dev/cosign/signing/signing_with_containers/.

7. Open Policy Agent (OPA), "Policy Language (Rego)," 2024. Available: https://www.openpolicyagent.org/docs/policy-language.
8. SPIFFE Project, "SPIFFE Specification," 2018. Available: https://spiffe.io/docs/latest/spiffe-about/overview/.
9. OpenTelemetry, "Logs Specification," 2024. Available: https://opentelemetry.io/docs/specs/otel/logs/.
10. Kubernetes Documentation, "Network Policies," 2024. Available: https://kubernetes.io/docs/concepts/services-networking/network-policies/.
11. Argo CD, "Sync Waves," 2025. Available: https://argo-cd.readthedocs.io/en/stable/user-guide/sync-waves/.
12. GitHub Docs, "Workflows and Actions Reference," 2025. Available: https://docs.github.com/en/actions/reference/workflows-and-actions.
13. Ansible Documentation, "Using playbooks," 2024. Available: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html.
14. HashiCorp, "Terraform Documentation," 2024. Available: https://developer.hashicorp.com/terraform/docs.
15. Red Hat, "Red Hat OpenShift GitOps Documentation," 2024. Available: https://docs.redhat.com/en/documentation/red_hat_openshift_gitops/.
16. D. Notaro, I. Cardoso, and M. Gerndt, "A Survey of AIOps Methods for Failure Management," ACM Transactions on Intelligent Systems and Technology, vol. 12, no. 6, 2021. DOI: 10.1145/3483424.
17. NetManAIOps, "AIOps Challenge 2020 Dataset and Benchmarks," 2020. Available: https://github.com/NetManAIOps/AIOps-Challenge-2020-Data.
18. UCI Machine Learning Repository, "Incident Management Process Enriched Event Log," 2019. DOI: 10.24432/C57S4H.
19. Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network," in Proc. ACM SIGKDD, 2019. DOI: 10.1145/3292500.3330672.
20. H. Ren, B. Xu, Y. Wang, C. Zheng, S. Yang, and D. Zhang, "Time-Series Anomaly Detection Service at Microsoft," in Proc. ACM SIGKDD, 2019.
21. Y. Yao, J. Zhao, Z. Xu, et al., "ReAct: Synergizing Reasoning and Acting in Language Models," arXiv:2210.03629, 2022.
22. Q. Wu, G. Bansal, J. Zhang, et al., "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework," arXiv:2308.08155, 2023.