International Journal of Advanced Engineering, Management and Science (IJAEMS)
Peer-Reviewed Journal
ISSN: 2454-1311 | Vol-12, Issue-3; May-Jun, 2026
Journal Home Page: https://ijaems.com/
DOI: https://dx.doi.org/10.22161/ijaems.123.22

C3P: A Continuous Compliance Control Protocol for Regulated Software Delivery

Paul Gresham

Paul Gresham Advisory LLC, New York, USA
Email: paul@paulgresham.com
ORCID: 0009-0005-2975-1187

Received: 29 Apr 2026; Received in revised form: 26 May 2026; Accepted: 29 May 2026; Available online: 05 Jun 2026

Abstract *— This paper presents C3P (Continuous Compliance Control Protocol), a design-science research contribution that establishes end-to-end traceability, integrity, and provenance across the full software delivery lifecycle in highly regulated environments. The framework addresses a structural conflict inherent in modern software engineering: CI/CD pipelines optimize for speed and automation, while traditional compliance models — maker-checker controls, four-eyes approvals, point-in-time attestations — assume slow, sequential delivery. This conflict produces three endemic failure modes: traceability gaps, integrity gaps, and evidence gaps. C3P resolves this by embedding compliance directly into the delivery pipeline as a continuous, verifiable output, introducing the concept of CI/CD/CC (Continuous Integration / Continuous Delivery / Continuous Compliance). The paper describes the framework's minimum artifact graph, control-plane separation model, evidence pack mechanism, and policy-as-code gate architecture. A working reference implementation built on widely available platform tooling demonstrates the framework's feasibility. The framework is informed by the author's decades of practitioner experience delivering software in globally regulated financial institutions, where precursor frameworks of this design were endorsed by independent internal audit functions and adopted at enterprise scale.*

Keywords *— **audit governance, CI/CD, continuous compliance, design science research, regulated software engineering, traceability***

I. INTRODUCTION

A. Background and Motivation

The software delivery landscape has undergone a fundamental transformation over the past two decades. Continuous Integration and Continuous Delivery (CI/CD) pipelines have become the standard mechanism for software deployment in both regulated and unregulated industries (Humble & Farley, 2010), and the underlying processes are now formalized at the industry level by IEEE Std 2675-2021 for DevOps build, package, and deployment (IEEE, 2021). These pipelines are designed to maximize deployment frequency, reduce lead time for changes, and improve system reliability (Forsgren et al., 2018). However, the compliance and audit requirements that govern regulated industries — particularly financial services, healthcare, and critical infrastructure — have not evolved at the same pace.

Traditional compliance frameworks rely on maker-checker controls, four-eyes approvals, manual sign-offs, and point-in-time attestations (U.S. Government Accountability Office, 2014). These mechanisms assume a sequential, waterfall-style delivery process where changes are planned, reviewed, approved, and deployed in discrete phases. Under high-frequency deployment models, these mechanisms become either impractical or counterproductive, creating a false choice between

delivery speed and compliance integrity (Saravia, 2018).

This structural conflict produces three predictable failure modes:

- Traceability gaps. It is difficult or impossible to prove which business requirement drove a specific code change, especially at high deployment frequency (Neale, 2019).
- Integrity gaps. It is difficult to prove that the code reviewed by a human is the same code that was built, tested, and deployed to production (Open Source Security Foundation, 2023).
- Evidence gaps. Compliance evidence is assembled retrospectively through manual processes rather than produced continuously as a durable output of the delivery pipeline (Fitzgerald & Stol, 2019).

These are not theoretical failures. They represent the root cause of recurring audit findings, regulatory censures, and compliance failures across major financial institutions (Basel Committee on Banking Supervision, 2013, 2023; Office of the Comptroller of the Currency, 2020). The problem is particularly acute in Global Systemically Important Financial Institutions (G-SIFIs), where the scale of software delivery — thousands of applications and hundreds of deployments per day — renders manual compliance mechanisms unsustainable (Financial Stability Board, 2009).

B. Research Contribution

This paper presents C3P (Continuous Compliance Control Protocol), an original control framework that addresses the structural conflict between CI/CD and regulatory compliance. The contribution is threefold:

- A minimum artifact graph that links business change records through source code, build artifacts, test evidence, and production deployment — a chain of custody that no existing compliance framework covers in full.
- An evidence pack mechanism — a standardized, per-release compliance bundle produced automatically by the pipeline, rather than assembled after the fact.
- A working reference implementation demonstrating that the framework's control objectives can be enforced through automated hard gates and role segregation using widely available platform tools.

The framework is presented as a control framework, not a prescriptive Software Development Life Cycle (SDLC). It defines what must be true for governance, auditability, integrity, and traceability — not how engineering work must be performed. Teams retain their preferred engineering process while satisfying non-negotiable control objectives.

C. Paper Structure

The remainder of this paper is structured as follows. Section II reviews related work in compliance frameworks, supply chain security, and continuous compliance. Section III presents the research methodology. Section IV describes the C3P framework in detail. Section V presents the working implementation and supporting practitioner evidence. Section VI discusses the findings and their implications. Section VII concludes.

II. RELATED WORK

A. Traditional Compliance Frameworks

The foundation of regulated software delivery compliance lies in frameworks such as SOX Section 404, IT General Controls (ITGC), and ISO/IEC 27001 (Sarbanes-Oxley Act, 2002; International Organization for Standardization, 2022). These frameworks were designed for waterfall and sequential delivery models and rely on manual controls, periodic audits, and retrospective evidence gathering. While effective for low-frequency deployment environments, they break down under the velocity requirements of CI/CD (Saravia, 2018).

The Sarbanes-Oxley Act of 2002 established the requirement for internal controls over financial reporting, creating a compliance infrastructure that has been adapted to software delivery but not fundamentally re-architected for it (Sarbanes-Oxley Act, 2002). NIST SP 800-53 provides a comprehensive catalog of security and privacy controls but does not address the continuous delivery dimension (National Institute of Standards and Technology, 2020).

B. Supply Chain Security Frameworks

The software supply chain security space has seen significant development in recent years. SLSA (Supply-chain Levels for Software Artifacts) provides

a framework for establishing artifact provenance through a maturity model (Google Cloud, 2021; Open Source Security Foundation, 2023). While SLSA addresses integrity and provenance for build artifacts, it does not cover the full compliance chain — from business change through requirement, code review, build, test, and deployment.

Similarly, the Software Bill of Materials (SBOM) movement, supported by NIST SP 800-161 Rev. 1 and Executive Order 14028 (2021), provides component-level transparency but does not address the governance and audit requirements of regulated environments (National Institute of Standards and Technology, 2024). The Secure Software Development Framework (SSDF), published as NIST SP 800-218, provides a structured approach to secure software development but is oriented toward security rather than compliance (National Institute of Standards and Technology, 2022).

C. Continuous Compliance in DevOps

The concept of embedding compliance into DevOps pipelines has been explored in academic literature. Saravia (2018) introduced compliance engineering as a discipline for integrating compliance into DevOps, identifying patterns for continuous compliance checking. Neale (2019) explored compliance automation in cloud environments, demonstrating that policy-as-code can enforce compliance continuously.

Ramaj et al. (2022) conducted a systematic literature review on compliance in DevSecOps, identifying the key challenges of maintaining compliance while adopting DevOps practices. Their findings confirm that the fundamental tension between speed and control persists and that automated compliance mechanisms are the primary solution path.

Sarne et al. (2024) examined the operational challenges of continuous compliance in large-scale engineering organizations, identifying that the primary barrier is not technical but organizational — specifically, the need for control-plane separation and role boundaries.

Zakharchenko (2026) introduced the Continuous Compliance Framework (CCF), a data-engineering reference architecture that treats compliance as a first-class, computable system property by combining declarative policies-as-code, standardized evidence collection, and cryptographically verifiable attestations recorded through transparency logs. CCF and the C3P framework presented in this paper address overlapping problems through complementary lenses: CCF as a data architecture for queryable, time-indexed compliance evidence, and C3P as a control framework defining what evidence must exist, who must produce it, and what organizational separations make the resulting evidence credible.

D. Gaps in Existing Literature

No existing framework addresses the full compliance chain in regulated software delivery. Traditional frameworks lack the automation and continuity required for CI/CD. Supply chain security frameworks lack governance, audit, and regulatory dimensions. DevOps compliance research has identified the problem but has not produced a complete, implementable framework. C3P fills this gap.

III. RESEARCH METHODOLOGY

This paper employs a design science research (DSR) methodology (Hevner et al., 2004), which is appropriate for contributions that create and evaluate artifacts designed to solve identified organizational problems. The DSR process follows the six-phase model:

- Problem identification. The structural conflict between CI/CD and compliance in regulated environments was identified through three decades of practitioner experience in financial services technology delivery, including engagements across multiple G-SIFIs.
- Objectives definition. The framework must establish end-to-end traceability, integrity, and provenance; produce audit-grade evidence continuously; and be implementable using widely available platform tools.
- Design and development. The C3P framework was designed as a control framework (not an SDLC) with a minimum artifact graph, control-plane separation model, and evidence pack mechanism.

- Demonstration. A working reference implementation was constructed using GitHub for hard access gates, role segregation, and pipeline enforcement.
- Evaluation. The framework was evaluated against its control objectives in reference implementation, and against the operational evidence of precursor frameworks of the same design that were independently reviewed and endorsed by internal audit functions at major financial institutions.
- Communication. This paper presents the framework, its design rationale, implementation evidence, and implications for future research.

The design science approach is particularly appropriate because C3P is not a theory to be tested but an artifact to be evaluated for utility in solving a real-world problem (Venable, 2011).

IV. THE C3P FRAMEWORK

A. Control Objectives

A Continuous Compliance control framework must satisfy four load-bearing objectives. These are not minimal or simplifying assumptions — each is non-negotiable and each cascades into substantive governance requirements when applied at scale.

Objective 1: Traceability. Every deployment can be traced to a specific, authorized business change. Every code change can be traced to an approved requirement and change record.

Objective 2: Integrity and Provenance. The code that is reviewed is the code that is built and deployed. Build and deploy processes are reproducible. Artifacts and approvals are attributable to specific identities (non-repudiation).

Objective 3: Immutable History (Tamper-Evidence). Change records and requirement references are retained as journaled versions — edits create new versions. History cannot be silently rewritten; deletions follow records-management rules (Purvis et al., 2009).

Objective 4: Auditability by Default. Evidence is produced continuously and can be queried. Exceptions are explicit, bounded, and reviewable.

B. The Minimum Artifact Graph

At its core, C3P maintains a verifiable link between the following artifacts: a Business Change Record (BCR) capturing an approved business need with defined scope, risk classification, and owner; Requirements describing acceptance criteria, controls impacted, and test expectations derived from the BCR; Code Changes comprising commits and pull requests linked to the BCR and requirements; Review Attestations recording reviewer identity and approval timestamps; a CI/CD Pipeline Definition versioned in the repository and tied to repository history; a Build Artifact (or cryptographic digest) produced from the reviewed commit; Test Evidence comprising automated test results, coverage signals, and regression pack compliance; and a Release/Deployment Record capturing what artifact was deployed, where, when, and by whom or what system.

This artifact graph forms a continuous chain of custody, in the sense established by ISO/IEC 27037:2012 for digital evidence (International Organization for Standardization, 2012). Each link in the chain is cryptographically or procedurally verifiable. The graph is not a process — it is a set of control objectives that any SDLC can satisfy.

C. Control-Plane Separation

A critical design principle of C3P is the separation of control-plane assets from day-to-day development discretion (Gkioulos & Katsaros, 2017). The authoritative systems — repository, CI/CD runners, artifact stores, and their configuration — must be operated by functions not aligned to the application team. The central source code repository is managed outside application teams; only code from the central repository is eligible for build, test, and deployment, and source history cannot be rewritten. The central issue and task system is also managed outside application teams, with story and requirement history retained as journaled versions and no silent edits. CI/CD runners and pipeline governance are similarly managed outside application teams: pipelines are configured as code and versioned, and developers cannot bypass runners or hand-carry artifacts. The artifact repository is managed outside application teams, artifacts are immutable, and only CI/CD pipelines can publish artifacts. Taken together, these separations ensure that no single human has permission to break the C3P chain. This is

the organizational analogue of cryptographic tamper-evidence.

D. The Evidence Pack

For each release, C3P generates a compact evidence pack containing the BCR identifier and owner; requirement references and acceptance criteria; risk classification and impacted controls; pull request and commit identifiers; code review records and attestation timestamps; pipeline definition version and build identifier; artifact digest; test run identifiers and summarized outcomes; deployment environment, timestamp, and authorizing identity; and any exceptions, with approvals and expiry.

The evidence pack is produced automatically by the pipeline. It is not assembled retrospectively. This is the key operational difference between C3P and traditional compliance: evidence is a by-product of delivery, not a separate activity.

E. Policy-as-Code Gates

C3P encodes control checks into the CI/CD pipeline as policy-as-code, enforcing required BCR and requirement links before merge, required reviewers based on risk classification, blocked merges when acceptance criteria are missing, required test suites for promotion, and blocked deployments when evidence is incomplete. These gates are hard: they cannot be bypassed by individual developers or even team leads. Only the control-plane function, separated from the application team, can modify gate policies.

V. IMPLEMENTATION AND EVALUATION

A. Working Reference Implementation: Crisis Monitor

A working reference implementation of C3P has been constructed and is publicly available as the Crisis Monitor project (Gresham, 2026). The repository is hosted at https://github.com/PGALLC/crisis_monitor. This implementation demonstrates that the framework's control objectives can be achieved using widely available, commercially supported platform tools (GitHub, Docker, Kubernetes) without any custom infrastructure or proprietary compliance tooling.

The implementation uses a five-actor segregation-of-duties model. The framework treats the Business Change Record and its requirements as an immutable upstream input to the engineering platform: the Coder consumes BCRs and requirements authored by the Product Owner and cannot create or self-approve them. The table below illustrates one expression of this separation in commodity tooling; equivalent separations can be expressed under other platform conventions.

Table 1: Five-actor model — roles, permissions, and restrictions.

Role	Responsibility	Permissions	Restrictions
Product Owner	Authors and owns the BCR and requirements; defines acceptance criteria and risk classification	Read/write in the issue / BCR system; read on the repository	Cannot write code, cannot merge PRs, cannot alter CI/CD workflows, cannot deploy
Coder	Writes application code and tests	repo scope (Classic PAT)	Cannot merge, cannot deploy, cannot alter CI/CD workflows; no workflow scope
Reviewer	Independent code review and PR approval	Contents (R/W), Pull Requests (R/W)	Cannot write code, cannot deploy, cannot approve own PRs
Platform Engineer	CI/CD pipeline and infrastructure	Full repo admin, workflow scope	Must follow PR rules on main; cannot touch application code
SRE	Production deployment gatekeeper	Read access to repo, Environment approval	Cannot write code, cannot merge PRs

As one example of how commodity tooling can satisfy the linkage requirement, the Crisis Monitor implementation uses GitHub's native issue-closure semantics: pull requests reference BCR Issues via the keyword Closes #N in their description, and merging the pull request automatically closes the referenced

Issues and records the closing commit SHA against each one. This produces an immutable, automatically generated bidirectional link between the deployed artifact and its authorizing requirements, without manual evidence gathering. C3P does not mandate this specific mechanism; any platform convention that produces an equivalent immutable link satisfies the control objective.

The implementation satisfies all C3P control objectives:

Table 2: Reference implementation against C3P control objectives.

Criterion	Requirement	Met?
Traceability	Full chain from BCR to deployment	Yes — PRs reference Issues via Closes #N; merge auto-closes the Issue and records the closing commit SHA, producing an immutable bidirectional link
Integrity	Reviewed code = built code = deployed code	Yes — branch protection prevents direct pushes; pipeline only builds from main
Provenance	Cryptographic or procedural verification at each link	Yes — commit SHA tags on container images; pipeline runs are immutable platform logs
Tamper-evidence	Immutable history with journaled versions	Yes — Git history cannot be rewritten; audit logs are immutable
Auditability	Evidence produced continuously, queryable	Yes — C3P evidence pack generated automatically per deployment
Implementability	Achievable with widely available tools	Yes — GitHub (repository, PRs, workflows, environments, container registry)

B. Practitioner Evidence

The C3P framework is informed by three decades of practitioner experience delivering software in globally regulated financial institutions. Precursor frameworks built on the same control principles — traceability, integrity and provenance, immutable history, and continuous evidence — were deployed at scale in major financial institutions and were independently reviewed and endorsed by internal audit functions. These engagements predate the current generation of AI tooling, and the corresponding deployments are reported here as practitioner experience rather than as a formal case study.

The transition from these precursor practices to C3P involved formalization — moving from internal practice to a formally documented control framework with defined control objectives, artifact graph, and evidence pack specification; generalization — removing organization-specific dependencies to create a portable framework applicable across industries and regulatory regimes; and automation — specifying the framework in terms of policy-as-code gates and pipeline-integrated evidence generation, rather than manual processes.

C. Evaluation Criteria

The framework was evaluated against the following criteria:

Table 3: Framework evaluation against C3P criteria.

Criterion	Requirement	Met?
Traceability	Full chain from BCR to deployment	Yes
Integrity	Reviewed code = built code = deployed code	Yes
Provenance	Cryptographic or procedural verification at each link	Yes
Tamper-evidence	Immutable history with journaled versions	Yes
Auditability	Evidence produced continuously, queryable	Yes

Implementability	Achievable with widely available tools	Yes
Organizational fit	Compatible with existing SDLC practices	Yes

VI. DISCUSSION

A. Key Design Decisions

Control framework versus SDLC. A load-bearing design decision was to present C3P as a control framework rather than a prescriptive SDLC. An SDLC defines how engineering work is performed; a control framework defines what must be true for governance and auditability. This distinction allows teams to retain their preferred engineering process while satisfying non-negotiable control objectives.

Evidence pack as first-class artifact. The evidence pack is not an afterthought — it is a first-class output of the delivery pipeline, produced continuously and automatically. This inverts the traditional compliance model where evidence is assembled retrospectively (Fitzgerald & Stol, 2019).

Control-plane separation. The separation of control-plane assets from application teams is the organizational mechanism that makes the framework credible. Without this separation, the framework collapses into a self-certification exercise (Gkioulos & Katsaros, 2017).

B. Comparison with Related Work

Table 4: Comparison of C3P with related compliance and supply-chain frameworks.

Framework	Trace.	Integrity	Provenance	Full Chain	Audit Evidence
SOX / ITGC	Partial	Partial	No	No	Manual
NIST SP 800-53	Partial	Partial	No	No	Manual
SLSA	No	Yes	Yes	No	No
NIST SP 800-218	Partial	Yes	Partial	Partial	Partial
C3P	Yes	Yes	Yes	Yes	Automated

C3P is the only framework that addresses all four dimensions — traceability, integrity, provenance, and full chain — with automated evidence production.

C. Limitations

The reference implementation, while demonstrating feasibility, is based on a single platform (GitHub). The framework's principles are platform-agnostic, but the specific implementation details would need adaptation for other platforms such as GitLab, Azure DevOps, or Bitbucket.

The practitioner evidence underpinning the framework, while substantial, is not quantified in this paper. Precursor deployments demonstrated practical utility, but formal metrics — audit time reduction, evidence assembly time, audit finding resolution rate — were not systematically collected at the time. This represents an area for future empirical research.

The framework has not been formally evaluated in non-financial regulated industries such as healthcare and critical infrastructure. While the control objectives are industry-agnostic, the specific regulatory requirements differ significantly.

VII. CONCLUSION

This paper has presented C3P (Continuous Compliance Control Protocol), an original control framework that resolves the structural conflict between CI/CD and regulatory compliance in regulated software delivery environments. The framework establishes end-to-end traceability, integrity, and provenance through a minimum artifact graph, control-plane separation, and automated evidence pack generation.

A working reference implementation demonstrates that the framework's control objectives can be achieved using widely available platform tools. The framework is further informed by

extensive practitioner experience in globally regulated financial institutions, where precursor designs were endorsed by independent internal audit functions.

The framework is designed to be applicable beyond the specific operational context in which it was developed. The structural problems it addresses — traceability gaps, integrity gaps, and evidence gaps — are endemic to high-frequency software delivery in any regulated industry. Subsequent work will examine the application of the framework to AI-augmented and agentic software delivery, where the same control objectives become more critical, not less. As AI and agentic systems enter the delivery pipeline, emerging governance standards such as ISO/IEC 42001:2023 for AI management systems will intersect with C3P's control objectives, a direction we leave to future work.

ACKNOWLEDGMENTS

The author thanks the internal audit and engineering colleagues whose questions and challenges across many years of regulated delivery shaped the design principles formalized in this paper.

REFERENCES

[1] Basel Committee on Banking Supervision. (2013). Principles for effective risk data aggregation and risk reporting (BCBS 239). Bank for International Settlements. https://www.bis.org/publ/bcbs239.pdf

[2] Basel Committee on Banking Supervision. (2023). Progress in adopting the principles for effective risk data aggregation and risk reporting. Bank for International Settlements. https://www.bis.org/bcbs/publ/d501.pdf

[3] Exec. Order No. 14028, 86 Fed. Reg. 26633 (May 17, 2021). Improving the nation's cybersecurity. https://www.federalregister.gov/d/2021-10460

[4] Financial Stability Board. (2009). Guidance to assess the systemic importance of financial institutions, markets and instruments: Initial considerations. Financial Stability Board. https://www.fsb.org/2009/10/r_091107c/

[5] Fitzgerald, B., & Stol, K.-J. (2019). DevOps and software verification and validation: A systematic mapping study. Journal of Systems and Software, 155, 110489.

[6] Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps: Building and scaling high performing technology organizations. IT Revolution Press.

[7] Gkioulos, V., & Katsaros, I. (2017). Security governance in cloud computing environments. Computers & Security, 68, 1–15.

[8] Google Cloud. (2021, June 16). Introducing SLSA, an end-to-end framework for supply chain integrity. Google Cloud Blog. https://cloud.google.com/blog/products/application-development/google-introduces-slsa-framework

[9] Gresham, P. (2026). Crisis Monitor [Computer software]. GitHub. https://github.com/PGALLC/crisis_monitor

[10] Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75–105.

[11] Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley Professional.

[12] IEEE. (2021). IEEE standard for DevOps: Building reliable and secure systems including application build, package, and deployment (IEEE Std 2675-2021). Institute of Electrical and Electronics Engineers. https://standards.ieee.org/ieee/2675/6830/

[13] International Organization for Standardization. (2012). Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence (ISO/IEC 27037:2012). https://www.iso.org/standard/44381.html

[14] International Organization for Standardization. (2022). Information security, cybersecurity and privacy protection — Information security management systems — Requirements (ISO/IEC 27001:2022). https://www.iso.org/standard/27001

[15] International Organization for Standardization. (2023). Information technology — Artificial intelligence — Management system (ISO/IEC 42001:2023). https://www.iso.org/standard/42001

[16] National Institute of Standards and Technology. (2020). Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5). U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-53r5

[17] National Institute of Standards and Technology. (2022). Secure software development framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities (NIST SP 800-218). U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-218

[18] National Institute of Standards and Technology. (2024). Cybersecurity supply chain risk management practices for systems and organizations (NIST SP 800-

161 Rev. 1). U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-161r1

[19] Neale, S. (2019). Compliance as code: Automating regulatory compliance in the cloud. O'Reilly Media.

[20] Office of the Comptroller of the Currency. (2020). In the matter of Citibank, N.A.: Consent order (AA-EC-2020-65). U.S. Department of the Treasury. https://www.occ.gov/news-issuances/news-releases/2020/nr-occ-2020-132.html

[21] Open Source Security Foundation. (2023). SLSA v1.0: Supply-chain levels for software artifacts. https://slsa.dev/spec/v1.0/levels

[22] Purvis, M., Rad, R., & Ragowsky, A. (2009). A review of audit trail research. Journal of Information Systems, 23(2), 121–150.

[23] Ramaj, X., Sánchez-Gordón, M., Gkioulos, V., Chockalingam, S., & Colomo-Palacios, R. (2022). Holding on to compliance while adopting DevSecOps: An SLR. Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '22), 1–10.

[24] Sarbanes-Oxley Act of 2002, Pub. L. No. 107–204, 116 Stat. 745 (2002).

[25] Saravia, E. (2018). Compliance engineering: Integrating compliance into DevOps. IEEE Software, 35(4), 78–85.

[26] Sarne, J., Sánchez-Gordón, M., & Gkioulos, V. (2024). Industrial challenges in secure continuous development. Proceedings of the 46th International Conference on Software Engineering (ICSE '24).

[27] U.S. Government Accountability Office. (2014). Standards for internal control in the federal government (GAO-14-704G). U.S. GAO. https://www.gao.gov/products/gao-14-704g

[28] Venable, J. (2011). The role of artefacts in information systems design science research. Information Systems Journal, 21(1), 37–56.

[29] Zakharchenko, A. (2026). Integrating continuous compliance into DevSecOps pipelines: A data engineering perspective. Software, 5(1), 6. https://doi.org/10.3390/software5010006