INTERNATIONAL JOURNAL OF COMPUTER SCIENCE & INFORMATION SYSTEM

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

A Threat-Intelligence-Driven Adaptive Devsecops Architecture With Integrated Data Management And Privacy-By-Design Controls For Continuous Security

Anya L. Sharma

Department of Cybersecurity and Threat Modeling, University of Singapore, Singapore

Abstract: This article presents an integrative theoretical framework for Adaptive DevSecOps that tightly couples threat intelligence, dynamic data management for continuous retraining, privacy-by-design, and containerized CI/CD security to enable automated, risk-aware security decisions before code reaches production. The work synthesizes multiple strands of recent scholarship and practitioner guidance—threat intelligence mapping, blockchain-enabled intelligence lifecycle support, continuous retraining pipelines, DevOps metrics and project alignment, container security techniques, regulatory privacy constraints, and event-consistency tradeoffs in distributed systems—into a single, coherent model for securing modern cloud-native software delivery. The article first identifies gaps in contemporary DevSecOps practice, then defines methodological constructs for integrating automated threat feeds, retraining controls, and privacy-preserving data governance within CI/CD workflows. A detailed, text-based methodology describes architecture patterns, data flows, decision points, and governance controls; results are presented as conceptual outcomes and expected operational benefits, grounded in the referenced literature. The discussion examines theoretical implications, counter-arguments, limitations (including data quality, false positives, and compliance complexity), and avenues for future empirical validation. This contribution intends to guide researchers and practitioners toward provable, auditable, and privacy-respecting automation of security actions in the software delivery lifecycle.

Keywords: DevSecOps, threat intelligence, continuous retraining, container security, privacy-by-design, CI/CD, data governance.

INTRODUCTION

Modern software delivery operates under accelerating velocity, distributed teams, and pervasive cloud-native architectures. DevOps principles sought to align development and operations to accelerate delivery

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

and increase reliability; contemporary shifts demand that security—and specifically threat-aware automation—become an integral part of that alignment so that risk is managed continuously rather than retrofitted at the end of delivery pipelines (Bass, Weber, & Zhu, 2015; Alluri et al., 2020). Despite tool proliferation, organizations still face barriers to adopting advanced security tooling and integrating threat intelligence into pre-production workflows (Anthony, 2023). Simultaneously, advances in data science and emergent practices such as continuous retraining create opportunities to maintain machine-learning based detection capabilities aligned with the evolving threat landscape (Baumann et al., 2022; Syverson, 2019). The intersection of these trends gives rise to the research problem addressed here: how can DevSecOps pipelines be adapted and extended to automatically incorporate threat intelligence, fresh model retraining, and privacy-aware data governance to enable proactive, verifiable, and auditable mitigation actions before code reaches production?

Prior work offers critical but fragmented contributions. Ahmed et al. (2023) map blockchain and data science to cyber threat intelligence lifecycle phases—collection, processing, analysis, dissemination—suggesting ledger-based provenance and immutable sharing for certain intelligence flows. Baumann et al. (2022) investigate dynamic data management for continuous retraining, identifying the need for systematic data curation and lineage to allow models to remain effective as environments change. Brás (2021) analyzes container security in CI/CD pipelines, highlighting attack surfaces introduced by container images, registries, and runtime configurations. Regulatory frameworks and privacy requirements, such as the GDPR, impose constraints on what data can be collected and how it can be processed, thus coupling data governance to DevSecOps processes (European Parliament, 2018; Bhajaria, 2022). DevOps metrics and organizational alignment literature provide measurement and management lenses for evaluating pipeline changes (Kersten, 2018; Puppet Labs, 2019; Alluri et al., 2020). Together, these works suggest the feasibility of an integrated framework but do not present a single, cohesive model for the automation of threat-driven mitigation prior to production deployment. Recent work also explores the integration of threat intelligence with DevSecOps automation (Malik, 2025), reinforcing the timeliness of constructing a theory-driven, comprehensive framework.

This article contributes (1) a detailed theoretical framework—Adaptive Threat-Aware DevSecOps (ATAD)—that merges threat intelligence lifecycle automation, continuous retraining data management, container and CI/CD security controls, and privacy-by-design governance; (2) an operational methodology describing decision points, data flows, and governance constraints; and (3) an analytic discussion of benefits, trade-offs, and research directions. The framework is grounded in the extant literature listed in the References and aims to be both academically rigorous and actionable for practitioners seeking to make security decisions pre-production.

METHODOLOGY

The methodology describes a systems-level, text-based architecture and process for integrating threat intelligence into DevSecOps workflows with continuous retraining and privacy controls. No empirical

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

dataset is required by this conceptual study; instead the method synthesizes literature-derived design principles into a prescriptive architecture and process model suitable for implementation and later empirical evaluation.

Foundational principles and assumptions. The methodology rests on five foundational principles derived from the literature. First, security must be continuous and shift-left—embedded early in the lifecycle—consistent with DevSecOps ideals (Bass et al., 2015; Alluri et al., 2020). Second, threat intelligence must be actionable and integrated into automated decision-making rather than merely informing post-hoc analysis (Ahmed et al., 2023; Malik, 2025). Third, machine learning-based components that assist detection require continuous retraining with curated, lineage-aware data to remain effective (Baumann et al., 2022; Syverson, 2019). Fourth, operational artifacts such as container images and pipelines present unique security considerations that must be controlled within CI/CD (Brás, 2021; Sonar & Pedersen, 2020). Fifth, privacy and legal constraints (e.g., GDPR) must be embedded into data collection and processing flows—privacy-by-design—so that automated processes remain compliant and auditable (European Parliament, 2018; Bhajaria, 2022).

High-level architecture. The proposed architecture decomposes into four interacting subsystems: (A) Threat Intelligence Layer (TIL), (B) Data Management & Retraining Layer (DMRL), (C) CI/CD Enforcement & Container Security Layer (CCSL), and (D) Governance, Compliance & Metrics Layer (GCML). Each subsystem has specific responsibilities, inputs, outputs, and controls, described below and synthesized from the referenced studies.

A: Threat Intelligence Layer (TIL). TIL ingests diverse intelligence sources—open-source feeds, commercial feeds, internal telemetry—performing normalization, enrichment, scoring, and provenance tagging. Ahmed et al. (2023) emphasize mapping intelligence across the lifecycle: collection, processing, analysis, dissemination. Here, ingestion includes cryptographic provenance markers for each intelligence artifact, enabling traceability and selective sharing. Enrichment includes context matching to project-specific assets (e.g., container registries, known vulnerable libraries) so the intelligence becomes immediately actionable within pipeline gates (Ahmed et al., 2023; Malik, 2025).

B: Data Management & Retraining Layer (DMRL). DMRL provides curated datasets for ML components (detectors, anomaly models) and manages continuous retraining pipelines with robust lineage. Baumann et al. (2022) argue for dynamic data management to support continuous retraining; this methodology operationalizes that claim by specifying data tagging, retention, and validation policies: (1) tag data by source, asset, timestamp, and privacy sensitivity; (2) enforce retention windows and purging where required by regulation (European Parliament, 2018; Bhajaria, 2022); (3) validate data for label quality and distribution drift; (4) maintain lineage metadata for audit and rollback.

C: CI/CD Enforcement & Container Security Layer (CCSL). CCSL embeds security policy enforcement within CI/CD pipelines and controls container image provenance, vulnerability scanning, and runtime

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

benchmarking. Brás (2021) details container vulnerabilities in pipelines; Sonar & Pedersen (2020) provide cloud-native security practices. CCSL contains pipeline gates: static security checks (SAST), software composition analysis (SCA), container image scanning, policy-as-code evaluation, and an automated decision engine that consults TIL and DMRL outputs to recommend or enforce actions (block, fail build, add mitigations, or allow). Container signing and attestation mechanisms ensure image provenance and trust before deployment.

D: Governance, Compliance & Metrics Layer (GCML). GCML implements privacy-by-design policies, role-based access control for intelligence and model outputs, and measurement systems aligned with DevOps metrics (Kersten, 2018). GCML ensures that data used for retraining is GDPR-compliant where relevant (European Parliament, 2018; Bhajaria, 2022). It also defines incident response boundaries and escalation rules when automated gate outcomes exceed organizational risk thresholds (Anthony, 2023).

Data flows and decision logic. The methodology defines deterministic decision logic for pipeline gates. When a commit triggers a pipeline, the CCSL runs baseline checks. If SCA or SAST reports high-risk items, it triggers TIL queries: TIL determines whether any intelligence matches the vulnerable component or pattern (Ahmed et al., 2023; Malik, 2025). If matched, a risk score is computed using weighted factors: vulnerability severity, intelligence confidence, asset criticality, and exploitability. DMRL augments the decision by checking recent model outputs for related anomalies, adjusting thresholds to reduce false positives informed by continuous retraining metrics (Baumann et al., 2022). GCML verifies that any telemetry or sample data used for retraining complies with privacy constraints and that an audit trail is recorded. Final automated actions follow policy-as-code: block deployment, require remediation, or permit with compensating controls and documented exceptions.

Privacy-by-design and legal constraints. All data collection and model training must conform to GDPR and organizational privacy standards. Bhajaria (2022) prescribes runbooks for engineers that include data minimization, purpose limitation, and encryption. The methodology adopts these principles: telemetry used for intelligence correlation is pseudonymized when feasible; training datasets avoid unnecessary personal data; explicit consent or lawful basis must be recorded for datasets containing personal data; and data subject rights workflows are integrated into GCML.

Model life cycle, validation, and rollback. Drawing on Baumann et al. (2022), the methodology includes: (1) continuous monitoring for data and concept drift; (2) a retraining cadence triggered by drift detection or scheduled windows; (3) validation suites to compare new and incumbent models using holdout data, adversarial resilience checks, and explainability diagnostics (Syverson, 2019); (4) rollback mechanisms with versioned artifacts and lineage metadata.

Metrics and organizational alignment. The methodology maps to DevOps and security metrics (Kersten, 2018; Puppet Labs, 2019): deployment frequency, mean time to recover, change lead time, security false positive/negative rates, and the proportion of builds blocked by intelligence triggers. Alluri et al. (2020)

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

emphasize aligning development and operations teams; GCML supports cross-functional governance boards for escalation and metric-driven continuous improvement.

Implementation pattern: policy-as-code and attestation. Policy-as-code encodes risk thresholds and automated responses, enabling consistent enforcement and auditability. Attestation—container signing and SBOM (software bill of materials)—provides technical evidence of supply chain integrity (Brás, 2021; Sonar & Pedersen, 2020). The methodology recommends leveraging cryptographic attestations to connect pipeline artifacts with intelligence actions and audit logs.

Threat modeling and continuous red-team integration. Continuous threat modeling should feed pipeline policies; growth of security pattern languages (Hafiz et al., 2006) can be used to codify recurring attack and mitigation patterns. Periodic chaos and red-team exercises validate policy efficacy and help adapt intelligence-to-policy mappings.

Validation strategy. For empirical evaluation, implementers should instrument pipelines to collect the defined metrics, run controlled A/B tests comparing pipelines with and without ATAD components, and measure security outcomes (reduction in post-deployment incidents, false positive rates, compliance violations). Controlled experiments should measure both operational impact (lead time, build failures) and security efficacy (detected/prevented vulnerabilities), following Kersten's measurement principles (Kersten, 2018).

RESULTS

Because this is a conceptual and prescriptive work synthesizing extant literature, the results section presents expected outcomes, predicted operational benefits, and theoretical performance characteristics that follow from applying the methodology. Each result claim references the underlying literature that supports the expectation.

Enhanced early detection and prevention. Integrating threat intelligence into pipeline gates should materially increase early detection of exploitable components by providing contextualized signals that pure static analysis may miss (Ahmed et al., 2023; Malik, 2025). For example, a low-severity vulnerability flagged by SCA but correlated with high-confidence exploit chatter in the TIL should be treated as higher risk, prompting earlier remediation. This reduces the likelihood of vulnerable artifacts reaching production and aligns with the shift-left security principle (Bass et al., 2015).

Reduced model drift and improved detection accuracy. Dynamically managing retraining data with lineage and validation (DMRL) reduces model drift and preserves detection accuracy over time (Baumann et al., 2022; Syverson, 2019). By limiting retraining to curated, validated datasets and rejecting noisy or privacy-sensitive samples, models maintain higher precision and recall, lowering false alarms that otherwise desensitize operations teams.

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

Container supply-chain risk mitigation. Container image signing, SBOM enforcement, and registry hardening within CCSL reduce supply chain attack vectors associated with images and dependencies (Brás, 2021; Sonar & Pedersen, 2020). When combined with TIL-derived signals (e.g., known exploit campaign targeting a library present in images), the pipeline can enforce immediate blocking or image rebuilds with patched dependencies, preventing deployment of compromised artifacts.

Privacy-compliant intelligence usage. Embedding GDPR-aware policies in GCML prevents improper use of personal data in retraining and intelligence correlation, reducing legal risk and enabling auditable decisions (European Parliament, 2018; Bhajaria, 2022). For organizations operating across jurisdictions, policy-as-code enables jurisdiction-specific controls for data subjects' rights and data localization requirements.

Operational cost trade-offs and developer experience. A predictable consequence is increased pipeline complexity and potentially longer build times or higher false-blocking rates if policies are overly strict. However, well-aligned metrics and iterative tuning (Kersten, 2018; Alluri et al., 2020) allow organizations to optimize thresholds to balance security and velocity. Anthony (2023) highlights organizational barriers to tool adoption—this framework anticipates and addresses such barriers by emphasizing measurement, alignment, and transparent governance.

Auditability and provenance. Ledger-style provenance tracking for intelligence artifacts supports accountability and enables later forensic and compliance investigations (Ahmed et al., 2023). Attestation of pipeline artifacts and versioned model artifacts provide an auditable chain-of-custody from code commit through deployment, helping satisfy compliance audits and incident post-mortems.

DISCUSSION

This section interprets the proposed framework, contrasts it with potential alternatives, elaborates on theoretical implications, and candidly examines limitations and open challenges.

Theoretical implications. The ATAD framework reframes DevSecOps from a set of isolated tooling practices into a socio-technical system where data governance, automated decision-making, and organizational metrics are first-class citizens. By integrating threat intelligence directly into pre-production gates and coupling it with lineage-aware retraining, the framework operationalizes a closed-loop security lifecycle: intelligence informs policy, policy influences pipeline behavior, pipeline telemetry feeds retraining, and retraining refines intelligence-driven detection. This cyclic structure resonates with control-theoretic ideas of feedback and adaptation and provides a language for reasoning about stability (false alarm rates), responsiveness (time-to-mitigation), and cost (developer friction). The incorporation of blockchain-inspired provenance (Ahmed et al., 2023) suggests that immutable evidence can be instrumental in bridging technical decisions and legal accountability, creating a convergence between technical controls and regulatory evidence requirements.

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

Counter-arguments and responses. One plausible critique is that adding intelligence-driven automation increases operational complexity and may slow development cadence. This is valid but addressable: policy-as-code and careful metric tracking enable incremental deployment of mitigations, A/B testing, and threshold tuning to avoid catastrophic developer-facing disruptions (Kersten, 2018; Alluri et al., 2020). A second critique concerns the reliability of threat intelligence—feeds can be noisy, inconsistent, or adversary-poisoned. The framework mitigates this by requiring provenance tagging, confidence scoring, and cross-source validation (Ahmed et al., 2023) and by using machine-learning models that incorporate ensemble and adversarial validation techniques (Syverson, 2019). A third critique highlights privacy harms—intelligence correlation may require sensitive telemetry. The privacy-by-design mandates (Bhajaria, 2022; European Parliament, 2018) within GCML are essential: by default, minimize personal data usage, pseudonymize telemetry, and maintain consent/logs where legal basis requires it.

Limitations. The framework is conceptual and requires rigorous empirical validation. Key limitations include data quality dependency, heterogeneity of intelligence feeds, jurisdictional privacy complexity, and organizational change management. Data quality: intelligence feeds vary in fidelity and scope; without careful normalization and confidence estimation, the system may produce spurious blocks. Heterogeneity: different teams and projects may have unique asset mapping needs; the TIL must be extensible to domain contexts. Jurisdictional privacy: multi-national organizations must encode complex, sometimes conflicting, legal requirements into GCML; policy-as-code helps, but legal oversight is necessary. Organizational change: adoption requires cross-functional buy-in and potentially new roles (intelligence curators, model governance leads), echoing Anthony's (2023) findings on barriers to adoption; addressing these demands a strong governance model and executive sponsorship.

Opportunities for future work. Several research avenues arise. Empirical validation is paramount: controlled deployments that instrument and compare metrics—security outcomes, developer productivity, compliance incidents—will quantify the framework's operational trade-offs (Kersten, 2018). Research into adversary-resistant intelligence ingestion is needed to resist feed poisoning and deliberate misinformation (Ahmed et al., 2023). Another area is automated legal-policy synthesis—translating GDPR clauses into enforceable policy-as-code—reducing the translation burden between legal and engineering teams (European Parliament, 2018; Bhajaria, 2022). Investigating human factors—how developers perceive and respond to automated blocking, and how governance structures influence acceptance—will inform adoption strategies (Alluri et al., 2020; Anthony, 2023). Finally, exploring the potential for federated learning and privacy-preserving model updates could allow cross-organization collaboration on detection models without sharing raw telemetry, aligning with privacy and provenance aims (Baumann et al., 2022).

Practical recommendations. For practitioners considering ATAD-like adoption, recommended steps include: (1) start with a pilot focusing on high-criticality services; (2) implement a simple TIL ingestion pipeline with confidence scoring and provenance; (3) add one policy-as-code gate informed by TIL signals and measure developer impact; (4) implement DMRL principles for model retraining with lineage

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

metadata; (5) ensure GCML captures legal constraints and implements data minimization; (6) iterate policies using DevOps metrics to find a balance between security and velocity (Kersten, 2018). Emphasize transparency and developer education—explain why actions occur and how to remediate—to reduce resistance (Alluri et al., 2020; Anthony, 2023).

Relationship to existing frameworks. ATAD complements and extends the NIST Cybersecurity Framework (NIST, 2018) by specifying concrete architectural and process patterns for the "Detect" and "Respond" functions within software delivery pipelines. It also operationalizes AWS and cloud-native security guidance through container attestation and SBOM practices (Sonar & Pedersen, 2020). Where previous work focuses on components (e.g., container security or continuous retraining), ATAD integrates them to form a coherent lifecycle approach (Brás, 2021; Baumann et al., 2022). The framework is likewise compatible with DevOps metrics and organizational alignment principles (Kersten, 2018; Alluri et al., 2020), enabling measurable governance.

CONCLUSION

This article presents an integrative, literature-grounded framework—Adaptive Threat-Aware DevSecOps (ATAD)—that prescribes how threat intelligence, dynamic data management for continuous retraining, container and CI/CD enforcement, and privacy-by-design governance can be combined to automate risk-aware security decisions before code reaches production. The approach addresses contemporary gaps by making intelligence actionable within pipeline gates, ensuring ML components remain effective through curated retraining, and embedding privacy and compliance controls into automated processes. While promising, the framework requires empirical validation to quantify benefits and optimize trade-offs between security and development velocity. This research agenda spans technical, legal, and human factors and invites collaborative studies that instrument and measure ATAD deployments. The integration of provenance, policy-as-code, and continuous retraining creates an auditable, adaptive security lifecycle that aligns with modern software engineering practice and regulatory expectations. By doing so, the framework offers a path toward more resilient, privacy-respecting, and accountable software delivery in the face of dynamic cyber threats.

REFERENCES

1. Ahmed, I., Mia, R., & Shakil, N. A. F. (2023). Mapping blockchain and data science to the cyber threat intelligence lifecycle: Collection, processing, analysis, and dissemination. Journal of Applied Cybersecurity Analytics, Intelligence, and Decision-Making Systems, 13(3), 1-37.
2. Alluri, R. R., Venkat, T. A., Pal, D. K. D., Yellepeddi, S. M., & Thota, S. (2020). DevOps Project Management: Aligning Development and Operations Teams. Journal of Science & Technology, 1(1), 464-487.
3. Anthony, R. T. (2023). Barriers to Adoption of Advanced Cybersecurity Tools in Organizations. Capitol Technology University.

Volume10 Issue09, September-2025, pg. 21-29

Published Date: - 30-09-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

4. Baumann, N., Kusmenko, E., Ritz, J., Rumpe, B., & Weber, M. B. (2022, October). Dynamic data management for continuous retraining. In Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (pp. 359-366).
5. Bass, L., Weber, I., & Zhu, L. DevOps: A Software Architect's Perspective. Addison-Wesley Professional, 2015.
6. Bhajaria, N. (2022). Data Privacy: A runbook for engineers. Simon and Schuster.
7. Brás, A. E. R. (2021). Container Security in CI/CD Pipelines (Master's thesis, Universidade de Aveiro (Portugal)).
8. Hafiz, M., Adamczyk, P., & Johnson, R. (2006). Growing a Security Pattern Language for Web Applications. Proceedings of the 13th Conference on Pattern Languages of Programs.
9. Kersten, M. (2018). DevOps Metrics: Quantifying the Performance of the DevOps Lifecycle. IEEE Software, 35(6), 94–97.
10. Malik, G. (2025). Integrating Threat Intelligence with DevSecOps: Automating Risk Mitigation before Code Hits Production. Utilitas Mathematica, 122(2), 309–340.
11. M. Sonar & O. Pedersen. (2020). Cloud-Native Security: Advancing DevSecOps Capabilities with AWS (AWS Whitepaper).
12. NIST. (2018). Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1.
13. Puppet Labs. (2019). State of DevOps Report.
14. Syverson, P. (2019). AI in Security Operations: Leveraging Machine Learning for Better Detection. IEEE Security & Privacy Magazine, 17(5), 22–29.
15. European Parliament. (2018). General Data Protection Regulation (GDPR).