pISSN:2349-5707 Volume:12, Issue 11, November-2025 eISSN:2349-5715

SECURITY-BY-DESIGN IN INTELLIGENT CYBER-PHYSICAL SYSTEMS: AN AI-ENHANCED ADAPTIVE DEVSECOPS ARCHITECTURE

Zheng Li
School of Software Engineering and DevSecOps Pipelines, Shanghai Jiao Tong University, China

ABSTRACT: Background: The convergence of DevOps practices with security (DevSecOps) has become a necessary evolution to secure modern cloud-native and cyber-physical systems; yet the integration of artificial intelligence (AI/ML), threat intelligence automation, and Security-by-Design into DevSecOps pipelines remains immature (Myrbakken & Colomo-Palacios, 2017; Carter, 2017; Malik, 2025).
Objective: This paper synthesizes evidence from literature and professional studies to advance an integrative conceptual framework—Adaptive DevSecOps for Intelligent Cyber-Physical Systems (ADICS)—that formalizes how AI/ML, automated threat intelligence, and model-based security assurances interoperate within continuous delivery for IoT, smart grid and GNSS-dependent systems (Yan et al., 2012; Thombre et al., 2017; Pramanik et al., 2017).
Methods: We perform an analytic synthesis of empirical studies, systematic reviews, and methodological contributions (Erich et al., 2017; Ahmed et al., 2021; Casola et al., 2024), mapping security controls to pipeline stages and describing AI/ML roles in verification and runtime monitoring (Cankar et al., 2023; Nebojsa Djosic et al., 2020). We then develop a prescriptive, text-based methodology for integrating threat intelligence automation and Security-by-Design SLAs into CI/CD.
Results: ADICS articulates (1) a layered threat model for cyber-physical contexts, (2) an AI/ML lifecycle integrated into build, test, and monitoring stages, and (3) automated risk mitigation flows that can block or quarantine artifacts before production (Malik, 2025; Bromberg & Gitzinger, 2020). We identify trade-offs between automation, explainability, and governance and provide operational controls for each trade-off.
Conclusions: Integrating AI/ML and real-time threat intelligence into DevSecOps can materially raise security assurance for intelligent cyber-physical systems, but success requires model-based security SLAs, continuous verification, and governance frameworks that balance automation with human oversight (Casola et al., 2020; Casola et al., 2024). Recommendations and a research agenda are provided.

Keywords: DevSecOps; AI/ML; Threat Intelligence; Security-by-Design; CI/CD; Cyber-Physical Systems.

INTRODUCTION

The last decade has seen an accelerating shift from siloed software development and operations toward integrated, continuous delivery practices known as DevOps (Erich, Amrit, & Daneva, 2017). DevSecOps extends this cultural and technical integration by embedding security practices and tools throughout the delivery lifecycle rather than treating security as a downstream gate (Myrbakken & Colomo-Palacios, 2017; Carter, 2017). The promise of DevSecOps is compelling: earlier discovery of vulnerabilities, faster remediation, and reduced risk at runtime. However, modern information systems increasingly operate in cyber-physical contexts—Internet of Things (IoT), smart grids, and GNSS-dependent infrastructures—where safety and physical consequences of cyber incidents raise the stakes (Yan et al., 2012; Thombre et al., 2017; Pramanik, Pal, & Choudhury, 2017). These environments expose new threat surfaces, demand real-time response, and require that security assurances accompany not only code but also deployed models and device firmware.

Simultaneously, advances in artificial intelligence and machine learning (AI/ML) present both challenges

and opportunities for security in the DevSecOps domain. AI/ML models power intelligent services and device decisioning, and their unique failure modes (data drift, adversarial examples, model poisoning) create an urgency to provide model-centric security controls (Huang & Rust, 2018; Guzman Camacho, 2024). Research and practitioner reports show early but fragmented approaches to combining AI/ML with DevSecOps: microservices for ML evaluation, ML-assisted testing, and runtime anomaly detection are emerging but lack standardized integration patterns (Bromberg & Gitzinger, 2020; Cankar et al., 2023).

There is therefore a clear gap: while DevSecOps practices, cloud computing reference models, and individual AI/ML security techniques are well documented (Mell & Grance, 2011; Casola et al., 2024), there remains no cohesive, operational framework that binds threat intelligence automation, model governance, and Security-by-Design into end-to-end delivery pipelines specifically tailored for intelligent cyber-physical systems. This paper addresses that gap by proposing the Adaptive DevSecOps for Intelligent Cyber-Physical Systems (ADICS) framework and detailed methodology, synthesizing existing literature and recent practitioner advances (Ahmed et al., 2021; Malik, 2025; Casola et al., 2020). The contribution is twofold: a conceptual, evidence-based architecture and a prescriptive, stage-by-stage methodology that maps controls, automations, and governance to CI/CD lifecycle events.

METHODOLOGY

This work follows a rigorous analytic synthesis approach appropriate to interdisciplinary conceptual development (Erich et al., 2017; Ahmed et al., 2021). The method involves three sequential activities: literature synthesis, threat and capability mapping, and prescriptive methodology design.

Literature synthesis. We systematically read and categorized the provided references into domains: DevOps and DevSecOps practice (Myrbakken & Colomo-Palacios, 2017; Erich et al., 2017; Carter, 2017), cloud and infrastructure context (Mell & Grance, 2011; Ehrlich et al., 2017), AI/ML in service and security (Huang & Rust, 2018; Guzman Camacho, 2024; Cankar et al., 2023), cyber-physical system specifics (Yan et al., 2012; Thombre et al., 2017), and applied DevSecOps research including monitoring and model evaluation (Ahmed et al., 2021; Bromberg & Gitzinger, 2020; Nebojsa Djosic et al., 2020). Each document was abstracted for: (a) pipeline stage relevance; (b) security control recommendations; (c) AI/ML applicability; and (d) governance or SLA implications.

Threat and capability mapping. From the synthesis we extracted common threat vectors across cyber-physical deployments (device compromise, supply chain injections, GNSS spoofing, data integrity attacks) and mapped these to potential mitigations available via DevSecOps automation: static analysis, SBOM checks, model validation, runtime anomaly detection, and threat intelligence-driven policy enforcement (Thombre et al., 2017; Yan et al., 2012; Malik, 2025).

Methodology design. We iteratively constructed the ADICS methodology, ensuring alignment with Security-by-Design principles and model-based security assurances (Casola et al., 2020; Casola et al., 2024). For each CI/CD stage (code, build, test, deploy, run), we specified roles for AI/ML, automated threat intelligence, and decision points where human oversight must be enforced. The resulting methodology emphasizes automated pre-production mitigation (e.g., policy gates that block artifacts), continuous verification, and runtime controls.

Throughout, we prioritized traceability: each major claim and prescribed control is grounded in at least one source from the provided references so that assertions are evidence-based (Myrbakken & Colomo-Palacios, 2017; Ahmed et al., 2021; Malik, 2025).

RESULTS

The primary result is the ADICS framework and associated prescriptive methodology. The framework is conceptually layered and operationally actionable.

Layered threat model for cyber-physical systems. Drawing on GNSS risk analysis and smart grid threat taxonomies (Thombre et al., 2017; Yan et al., 2012), the model classifies threats into: (1) sensor and device layer threats (firmware compromise, device hijack); (2) edge and gateway threats (man-in-the-middle, protocol manipulation); (3) cloud and orchestration threats (misconfigured services, compromised CI/CD credentials); and (4) model and data threats (training data poisoning, model evasion). Each class maps to a set of DevSecOps controls: firmware signing and SBOM verification for device threats, mutual TLS and message authentication for edge threats, IAM hardening and CI secret scanning for cloud threats, and data governance and model validation for model threats (Mell & Grance, 2011; Casola et al., 2020; Cankar et al., 2023).

AI/ML lifecycle integrated into pipeline stages. We specify concrete roles for AI/ML in the pipeline. During build/test stages, automated model evaluation microservices (as proposed in DroidAutoML architectures) can run reproducible tests against baseline datasets and adversarial scenarios (Bromberg & Gitzinger, 2020). In pre-production, ML-driven static and dynamic analyses screen for anomalies and supply-chain issues (Cankar et al., 2023). In production, AI/ML supports continuous monitoring for behavioral anomalies indicative of attacks or drift (Nebojsa Djosic et al., 2020; Ahmed et al., 2021). Critically, we recommend a versioned, reproducible model registry and evaluation harness as a mandatory pipeline artifact (Guzman Camacho, 2024).

Automated threat-intelligence flows. Inspired by Malik (2025), ADICS integrates threat intelligence so that CI/CD pipelines can enact automated risk mitigations—such as blocking deployments where package signatures match active IoCs or where a build's provenance is suspect. Threat feeds are normalized, risk-scored, and mapped to automated policy actions. For example, if intelligence indicates active exploitation of a library present in a build, the pipeline will quarantine the artifact and open a triage workflow (Malik, 2025).

Security-by-Design SLAs and model-based assurance. ADICS prescribes that each service component and ML model is associated with measurable security SLAs—quantitative requirements such as acceptable false positive rates for anomaly detection or maximum acceptable data drift metrics (Casola et al., 2020; Casola et al., 2024). These SLAs integrate into release gating: artifacts that do not meet SLA thresholds must not progress to higher environments.

Operational trade-offs and governance. The framework identifies trade-offs: more aggressive automated blocking reduces time-to-remediation but risks false positives that delay releases; heavier runtime model inference for detection improves situational awareness but increases resource usage and latency. ADICS recommends governance controls—human-in-the-loop adjudication thresholds, explainability reports for ML decisions, and change logs—to balance automation with human oversight (Huang & Rust, 2018; Guzman Camacho, 2024).

DISCUSSION

This section interprets the ADICS results, examines implications, discusses limitations, and proposes a research and practice agenda.

Interpretation and theoretical implications. ADICS synthesizes organizational, technical, and model-

centric perspectives into a unified approach. From a socio-technical viewpoint, this aligns with DevSecOps' cultural emphasis on shared responsibility while expanding the remit to include model governance and real-time threat intelligence (Myrbakken & Colomo-Palacios, 2017; Erich et al., 2017). Theoretically, ADICS advances three claims: (1) security in cyber-physical systems must be modeled as emergent from continuous interactions between code, models, and infrastructure; (2) AI/ML can accelerate detection and verification when operationalized as part of the pipeline lifecycle rather than an afterthought; and (3) quantifiable Security-by-Design SLAs provide necessary constraints to translate abstract governance goals into enforceable pipeline checks (Casola et al., 2020; Casola et al., 2024; Guzman Camacho, 2024).

AI/ML as both asset and liability. Literature shows AI/ML brings new failure modes that must be defended (Huang & Rust, 2018). ADICS treats models as first-class artifacts: they have provenance, versioning, and test suites. The model evaluation microservice idea from Bromberg & Gitzinger (2020) becomes operationalized—models undergo adversarial robustness tests, concept drift checks, and interpretability audits before being promoted. This mitigates risks such as model poisoning and silent performance degradation (Cankar et al., 2023; Nebojsa Djosic et al., 2020). However, reliance on ML for detection introduces concerns about explainability and legal compliance; the framework therefore prescribes explainability artifacts and thresholded human oversight for high-impact decisions (Huang & Rust, 2018).

Threat intelligence automation: efficacy and pitfalls. Malik (2025) shows how threat intelligence can automate risk mitigation earlier in the pipeline. ADICS extends that by formalizing mappings between intelligence indicators and automated actions. This automation reduces dwell time for known threats, but RM/QA teams should be aware of indicator accuracy and feed provenance. False indicators can block benign deployments; hence ADICS specifies confidence scoring, multi-source corroboration, and human adjudication gates as mitigations.

Security-by-Design SLAs: operationalizing security requirements. Security SLAs convert policy into measurable criteria. Casola et al. (2020; 2024) propose model-based methodologies for assessing SLAs; ADICS operationalizes those ideas by requiring quantitative acceptance tests in CI, including resilience metrics for models and components. This transitions security from qualitative checklists to measurable objectives, aligning with modern SRE practice.

Limitations. The proposed framework is conceptual and based on literature synthesis rather than empirical field trials. While many of its components (model registries, CI gating, threat feeds) are feasible and have been piloted in industry, ADICS requires organizational maturity—cross-functional collaboration, automation investment, and legal/regulatory alignment—that may be infeasible for smaller teams (Erich et al., 2017; Ahmed et al., 2021). Additionally, the integration of multiple data sources and ML models raises privacy and compliance concerns not exhaustively addressed here; the implementation must respect jurisdictional constraints.

Implementation challenges. Several practical barriers exist. Establishing reliable model evaluation datasets for cyber-physical system behaviors is non-trivial; synthetic or simulated datasets risk not representing real operational conditions (Pramanik et al., 2017). Maintaining the integrity of threat intelligence (ensuring low false positive rates and provenance) demands careful curation and may require subscription to commercial feeds or community collaboration (Malik, 2025). Integrating SBOM (software bill of materials) checks, firmware signing, and device attestation into CI/CD pipelines requires coordination across device manufacturers and supply chains, a known challenge in IoT ecosystems (Yan et al., 2012).

Future research and practice agenda. To advance ADICS from concept to validated practice, we recommend a multi-pronged agenda:

1. Empirical field studies deploying ADICS components in smart grid pilot projects and GNSS-dependent systems to validate detection performance, false positive rates, and deployment latency impacts (Thombre et al., 2017; Yan et al., 2012).

2. Benchmarks and open datasets for model evaluation in cyber-physical contexts, including adversarial scenarios and drift conditions, to facilitate reproducible comparisons of model-hardening techniques (Bromberg & Gitzinger, 2020).

3. Governance frameworks that specify liability, audit trails, and explainability requirements for automated mitigation actions—especially where human safety could be affected (Huang & Rust, 2018).

4. Standardized threat-intelligence policy maps that translate IoCs and TTPs into pipeline actions with defined confidence thresholds, reducing ad-hoc rule creation (Malik, 2025).

5. Tooling and APIs to integrate firmware SBOM verification and device attestation into mainstream CI/CD systems, bridging supply chain and DevSecOps workflows (Mell & Grance, 2011).

CONCLUSION

The Adaptive DevSecOps for Intelligent Cyber-Physical Systems (ADICS) framework presented here synthesizes existing scholarship and practitioner work into a coherent methodology that binds AI/ML, threat intelligence automation, and Security-by-Design into continuous delivery lifecycles. The framework addresses the unique security needs of IoT, smart grid, and GNSS-dependent systems by making models first-class artifacts, automating mitigation via intelligence feeds, and enforcing quantifiable SLAs as release gates. While the framework is conceptually robust, its practical realization requires organizational commitment, high-quality data, and careful governance to balance automation benefits against potential harms from incorrect or opaque automated decisions. The roadmap provided can guide practitioners and researchers toward operational pilots and empirical validation that will refine and strengthen the framework. Ultimately, integrating AI/ML and real-time threat intelligence into DevSecOps is a promising route to achieving resilient, secure, and trustworthy intelligent cyber-physical systems—provided that the community invests in measurement, explainability, and cross-disciplinary governance.

REFERENCES

1. Myrbakken, H., & Colomo-Palacios, R. (2017). DevSecOps: A Multivocal Literature Review. Communications in Computer and Information Science, 17–29. https://doi.org/10.1007/978-3-319-67383-7_2

2. Thombre, S., Bhuiyan, M. Z. H., Eliardsson, P., Gabrielsson, B., Pattinson, M., Dumville, M., Fryganiotis, D., Hill, S., Manikundalam, V., Pölöskey, M., Lee, S., Ruotsalainen, L., Söderholm, S., & Kuusniemi, H. (2017). GNSS Threat Monitoring and Reporting: past, present, and a Proposed future. Journal of Navigation, 71(3), 513–529. https://doi.org/10.1017/s0373463317000911

3. Erich, F. M. A., Amrit, C., & Daneva, M. (2017). A qualitative study of DevOps usage in practice. Journal of Software, 29(6). https://doi.org/10.1002/smr.1885

4. Huang, M., & Rust, R. T. (2018). Artificial intelligence in service. Journal of Service Research, 21(2), 155–172. https://doi.org/10.1177/1094670517752459

5. Yan, Y., Qian, Y., Sharif, H., & Tipper, D. (2012). A survey on cyber security for smart grid communications. IEEE Communications Surveys & Tutorials, 14(4), 998-1010. https://doi.org/10.1109/surv.2012.010912.00035

6. Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

7. Carter, K. (2017). Francois Raynaud on DevSecOps. IEEE Software, 34(5), 93–96. https://doi.org/10.1109/ms.2017.3571578

8. Ehrlich, M., Trsek, H., Lang, D., Wisniewski, L., Wendt, V., & Jasperneite, J. (2017). Security concept for a cloud-based automation service. In VDI Verlag eBooks (pp. 151–152). https://doi.org/10.51202/9783181022931-151

9. Pramanik, P. K. D., Pal, S., & Choudhury, P. (2017). Beyond automation: the cognitive IoT. Artificial intelligence brings sense to the internet of things. In Lecture notes on data engineering and communications technologies (pp. 1–37). https://doi.org/10.1007/978-3-319-70688-7_1

10. Ahmed Bahaa, Ahmed Abdelaziz, Abdalla Sayed, Laila Elfangary, & Hanan Fahmy. (2021). Monitoring real time security attacks for IoT systems using DevSecOps: a systematic literature review. Information, 12(4), 154.

11. Yérom-David Bromberg & Louison Gitzinger. (2020). DroidAutoML: A Microservice Architecture to Automate the Evaluation of Android Machine Learning Detection Systems. In Distributed Applications and Interoperable Systems: 20th IFIP WG 6.1 International Conference, DAIS 2020, Proceedings (pp. 148–165). Springer-Verlag. doi:10.1007/978-3-030-50323-9_10

12. Nicolas Guzman Camacho. (2024). Unlocking the potential of AI/ML in DevSecOps: effective strategies and optimal practices. Journal of Artificial Intelligence General science (JAIGS), 3(1), 106–115.

13. Matija Cankar, Nenad Petrovic, Joao Pita Costa, Ales Cernivec, Jan Antic, Tomaz Martincic, & Dejan Stepec. (2023). Security in DevSecOps: Applying Tools and Machine Learning to Verification and Monitoring Steps. In Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion) (pp. 201–205). Association for Computing Machinery. doi:10.1145/3578245.3584943

14. Valentina Casola, Alessandra De Benedictis, Carlo Mazzocca, & Vittorio Orbinato. (2024). Secure software development and testing: A model-based methodology. Computers & Security, 137, Article 103639. doi:10.1016/j.cose.2023.103639

15. Valentina Casola, Alessandra De Benedictis, Massimiliano Rak, & Umberto Villano. (2020). A novel Security-by-Design methodology: Modeling and assessing security by SLAs with a quantitative approach. Journal of Systems and Software, 163, 110537. doi:10.1016/j.jss.2020.110537

16. Malik, G. (2025). Integrating Threat Intelligence with DevSecOps: Automating Risk Mitigation

before Code Hits Production. Utilitas Mathematica, 122(2), 309-340.

17. CASP. (2019). CASP Qualitative Checklist. https://casp-uk.net/wp-content/uploads/2018/01/CASP-Qualitative-Checklist-2018.pdf

18. Nebojsa Djosic, Bojan Nokovic, & Salah Sharieh. (2020). Machine Learning in Action: Securing IAM API by Risk Authentication Decision Engine. In 2020 IEEE Conference on Communications and Network Security (CNS) (pp. 1–4). doi:10.1109/CNS48642.2020.9162317