COLOMBO SCIENTIFIC PUBLISHING

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE & INFORMATION SYSTEM
E-ISSN: 2536-7919
P-ISSN: 2536-7900

PAGE NO: 63-70

Integrating DevSecOps and LLM-Enhanced Mobile Security: A Unified Framework for Automated Security Testing in Modern CI/CD Pipelines

Sergiu Metgher
Global Institute of Technology, International University

ARTICLE INFO

Article history:

Submission: November 01, 2025
Accepted: November 17, 2025
Published: November 30, 2025
VOLUME: Vol.10 Issue 11 2025

Keywords:
DevSecOps, CI/CD pipelines, automated security testing, mobile applications, LLM security, static analysis, dynamic analysis, compliance artifacts

ABSTRACT

With the accelerating shift toward rapid software delivery through continuous integration and continuous deployment (CI/CD) pipelines, the need for robust, automated security testing has never been more critical. Traditional security processes often lag behind the pace of development, leading to increased vulnerability exposure. The DevSecOps paradigm—integrating development, operations, and security—has emerged to address this gap by embedding security checks into CI/CD workflows. Simultaneously, the rise of mobile applications that incorporate large-language-model (LLM) functionalities introduces new privacy and security risks that demand automated, systematic testing. This paper synthesizes extant research on DevSecOps automation and extends it to propose a unified framework tailored for modern mobile applications enhanced by LLMs. Drawing on empirical and conceptual studies of automated scanning, static and dynamic analysis, heuristic algorithm-based testing, security compliance models, and barriers to adoption, we construct a comprehensive pipeline architecture. We argue that integrating heuristic-driven testing (via genetic algorithms), static and dynamic vulnerability detection, security compliance artifacts, and specialized LLM-privacy testing can significantly raise the security baseline while maintaining deployment velocity. We elaborate on architectural design, methodological considerations, potential challenges, and future directions, including governance, scalability, and human-in-the-loop oversight. This synthesis aims to guide both academia and industry toward more resilient, automated security practices in a landscape increasingly shaped by AI-enhanced mobile software.

INTRODUCTION

The software development lifecycle has undergone a profound transformation over the past decade. The traditional waterfall or sequential model has given way to more agile, iterative paradigms that demand rapid delivery while maintaining high quality. Concurrently, as software becomes embedded deeply in high-stakes contexts—financial, healthcare, defense, personal data repositories—the cost of security failures has amplified substantially. In recognition of this, many organizations have adopted the DevSecOps approach: integrating security into development (Dev) and operations (Ops), rather than treating it as an afterthought or separate silo (Jammeh, 2020; Abiola & Olufemi, 2023).

Under DevSecOps, security controls and testing are baked into the CI/CD pipeline, enabling continuous verification against vulnerabilities and misconfigurations, and accelerating feedback to developers (Hsu, 2019; Putra & Kabetta, 2022). However, despite advances, a variety of challenges remain: insufficient automation, reliance on static or dynamic testing alone, limited adaptability, difficulties in compliance verification, and human bottlenecks (Jones, 2023; Lorona, 2023).

Simultaneously, the world of mobile applications is rapidly evolving, now often embedding artificial intelligence components—including large language models (LLMs)—to provide advanced features such as natural-language interaction, content generation, summarization, and personalization. These capabilities bring new security and privacy risks, especially when deployed on resource-constrained mobile devices

with sensitive data (Security and Privacy Testing Automation for LLM-Enhanced Applications in Mobile Devices, 2025).

Despite these converging trends—DevSecOps automation and LLM-enhanced mobile apps—there is a surprising gap in the literature: no comprehensive framework currently addresses automated security testing for LLM-enhanced mobile applications within a DevSecOps CI/CD pipeline. Existing DevSecOps research largely focuses on traditional static and dynamic scanning, compliance checks, and vulnerability detection in classical codebases (Marandi et al., 2023; Thantharate & Anurag, 2023; Abiola & Olufemi, 2023; Putra & Kabetta, 2022). Meanwhile, work on LLM security and privacy testing—particularly for mobile contexts—is nascent and isolated (Security and Privacy Testing Automation for LLM-Enhanced Applications in Mobile Devices, 2025).

This paper seeks to bridge that gap by synthesizing the state-of-the-art in DevSecOps security automation and extending it to encompass LLM-enabled mobile applications. We propose a unified, modular framework for automated security testing, combining static analysis, dynamic testing, heuristic-based vulnerability search, compliance artifact generation, and specialized LLM privacy/stability testing. Our goal is to provide a blueprint that organizations can adopt to ensure robust security without sacrificing the agility afforded by iterative CI/CD workflows.

In what follows, we first present a detailed methodology describing how various security-testing techniques can be integrated in a coherent pipeline. Then we present a conceptual results section, describing the expected benefits, trade-offs, and hypothetical performance outcomes of the proposed architecture. The discussion delves into limitations, potential obstacles in adoption, governance and human-in-the-loop concerns, and avenues for future work. Finally, the conclusion summarizes our contributions and underscores the importance of this integration in modern software development.

METHODOLOGY

We propose a conceptual, modular architecture for a unified DevSecOps + LLM-mobile security pipeline. The following subsections describe each module, their interactions, and the rationale for inclusion—grounded in existing scholarship.

Architecture Overview

The overarching architecture centers on a CI/CD pipeline, typical of modern software engineering practices, extended with security-automation modules at multiple stages:

- Pre-commit static analysis
- Build-time heuristic-driven security testing
- Pre-deployment dynamic scanning and behavioral testing
- LLM-specific privacy and security validation (for mobile builds embedding LLMs)
- Compliance artifact generation
- Feedback loop for developers and security teams

This modular design allows flexibility—teams can enable or disable modules as needed, based on application type, risk profile, or regulatory context—while preserving a unified, coherent pipeline.

Pre-commit Static Analysis Module

Drawing inspiration from classical static code analysis tools and practices described in DevSecOps literature (Hsu, 2019; Putra & Kabetta, 2022), this module analyzes source code (or intermediate code) before it enters the shared repository. The goals are to detect coding patterns prone to vulnerabilities:

insecure API usage, deprecated libraries, hardcoded secrets, buffer overflows (for native code), misconfigurations, and insecure permissions (especially critical in mobile context).

This module leverages a suite of static analyzers: linters, taint-analysis tools, dependency scanners, and configuration checkers. By running early, the static analysis module helps prevent a large class of vulnerabilities before they propagate.

Advantages include near-instant feedback to developers, low overhead, and minimal resource requirements. However, static analysis alone cannot capture runtime behavior, environment-specific vulnerabilities, or interactions with external libraries—including embedded LLM frameworks or runtime models.

Build-time Heuristic-driven Security Testing Module

To address the limitations of static-alone analysis, we propose integrating a heuristic approach to security testing at build-time, inspired by the research of Thantharate & Anurag (2023). Their work on GeneticSecOps demonstrates how genetic algorithms can be leveraged to generate test sequences aimed at uncovering vulnerabilities—especially those that are context-dependent or arise from complex interactions.

In our framework, after compiling the application (including bundling any LLM models for mobile deployment), the build-time module initiates a heuristic-driven test generation engine. This engine operates akin to fuzz testing but uses genetic algorithms to evolve test inputs over successive generations, guided by vulnerability detection feedback (e.g., crash logs, exception traces, resource anomalies, privacy violation indicators).

The genetic algorithm-based engine would iterate through cycles of test input generation, execution (in an emulated or sandbox environment), assessment (vulnerability signals, privacy leaks, deviation from expected behavior), and selection of promising variants for further mutation and recombination.

This method leverages the strengths of heuristic search—in exploring large, complex input spaces with limited human guidance—and can reveal edge-case vulnerabilities that static or conventional fuzz testing might miss. It is especially useful when applications include large, opaque components like LLM libraries, which complicate whitebox analysis.

Pre-deployment Dynamic Scanning and Behavioral Testing Module

At this stage, the pipeline performs dynamic analysis of the application in an environment that closely mirrors production—on actual devices or device simulators for mobile contexts. This dynamic module is responsible for detecting vulnerabilities that only manifest at runtime: memory leaks, insecure network traffic, improper permission handling, insecure data storage, misuse of system APIs, and behavioral anomalies under load or under unusual input patterns.

Additionally, for LLM-enhanced mobile applications, dynamic testing includes stress-testing the integration of LLM components: validating resource management (CPU, memory), monitoring for unintended data leakage, ensuring secure communication with backend servers (if applicable), and verifying that no sensitive data is inadvertently processed or stored insecurely. Dynamic testing may involve automated UI-driven scenarios, background event simulations, and randomized or heuristic-driven interactions.

This module builds on the foundational approaches described by Marandi et al. (2023), who demonstrated the practical viability of integrating dynamic scanning into DevSecOps CI/CD pipelines. It ensures that runtime safety and security are validated before deployment, significantly reducing the likelihood of security incidents in production.

LLM-Specific Privacy and Security Validation Module

Given the proliferation of mobile applications embedding LLM functionality—for personalized content, chatbots, summarization, natural-language generation, etc.—there is a compelling need for specialized security and privacy validation of such integrations (Security and Privacy Testing Automation for LLM-Enhanced Applications in Mobile Devices, 2025).

This module integrates two sub-components:

● Privacy Leakage Detection: The system automatically generates inputs that attempt to elicit the LLM's handling of potentially sensitive user data. This includes boundary test cases for user-provided personal

information, structured or unstructured inputs that attempt to trigger data retention or output beyond expected scope, and adversarial prompts designed to coax the model into revealing training data or cached user data. The privacy module logs all outputs and flag any responses that may leak sensitive information, PII, or internally stored data.

● Behavioral Robustness and Safety Testing: This subcomponent subjects the LLM integration to adversarial or unexpected user behavior—such as malformed inputs, malformed encoding, rapid-fire sequences, resource stress, or concurrent requests—to test for crashes, model failures, or unintended side effects, including resource exhaustion, denial-of-service vulnerabilities, or unhandled exceptions that might compromise the application or device security.

Together, these ensure that embedding LLM functionality does not introduce novel vulnerability vectors that traditional static/dynamic scanning would miss. Given the novelty of LLM embedding in mobile apps, such specialized testing is essential (Security and Privacy Testing Automation for LLM-Enhanced Applications in Mobile Devices, 2025).

Compliance Artifact Generation Module

Even when security testing is automated, many organizations—especially those in regulated domains (finance, healthcare, defense)—require documentation and compliance artifacts: audit trails, evidence of scanning, reports of vulnerabilities found and remediated, versioned configuration snapshots, and compliance checklists.

The compliance artifact generation module automatically collects data produced by previous modules (static scan reports, dynamic analysis logs, heuristic test results, privacy leakage reports) and assembles them into structured artifacts—e.g., machine-readable scan result logs, human-readable summary reports, audit ready packages, and historical records.

This approach draws on the work by Bitra & Achanta (2021), who developed models for artifacts in DevSecOps to support security compliance. By automating artifact generation, organizations can ensure traceability, accountability, and readiness for audits without overburdening developers or security teams.

Feedback Loop and Developer/Security Team Integration

A critical dimension often underemphasized is the feedback loop—how findings are communicated back to developers, security engineers, or operations staff. Without meaningful and timely feedback, even an automated pipeline may have limited real-world impact (Jammeh, 2020; Lorona, 2023; Jones, 2023).

In our framework, after each pipeline run (or after failed security tests), detailed reports are generated and communicated to stakeholders. For developers, reports include actionable items: lines of code to remediate, misconfigurations, insecure API usage, or flagged privacy issues. For security teams or compliance officers, summary dashboards highlight risk levels, policy compliance status, and historical vulnerability trends. For project managers or executives, high-level risk assessments, compliance status, remediation progress, and deployment readiness indicators are generated.

By automating and structuring feedback, the pipeline encourages accountability, continuous improvement, and cross-functional collaboration—core tenets of successful DevSecOps adoption.

Governance, Configuration, and Customization Considerations

Given organizational diversity—ranging from small startups to large enterprises, across domains, regulatory regimes, and technology stacks—the proposed framework emphasizes configurability. Teams can tune modules: enable or disable heuristic testing, adjust privacy leakage thresholds, define compliance artifact formats, integrate human review gates, and customize reporting. Governance policies can specify mandatory modules (e.g., LLM privacy testing in sensitive contexts), optional modules, and override mechanisms.

Moreover, the architecture supports extensibility: as new security testing tools emerge (e.g., better static analyzers, advanced fuzzers, improved privacy testers), they can be plugged into the pipeline without redesigning the entire system. This modular, extensible nature helps future-proof the framework as both mobile applications and LLMs evolve.

RESULTS

Because this work is conceptual—drawing on existing literature rather than empirical data—we present expected results in terms of qualitative benefits, hypothetical quantitative improvements, and trade-offs. The goal is to articulate how adoption of the unified framework can enhance security posture, mitigate risk, and support compliance, while maintaining development velocity.

Anticipated Security Improvements

● Early Vulnerability Detection: By combining static analysis with heuristic-driven testing and dynamic scanning, the framework addresses a wide spectrum of vulnerabilities—ranging from coding flaws to runtime behavior anomalies and privacy leaks. Early detection (pre-commit and build-time) reduces the cost and impact of remediation.

● Reduced Attack Surface: For mobile applications embedding LLMs, the privacy validation module can detect and block inadvertent leakage of sensitive user data or model-triggered data retention vulnerabilities. This reduces exposure to data breaches, compliance violations, or privacy incidents.

● Comprehensive Auditability: Automatic compliance artifact generation ensures traceability and audit readiness, which is especially valuable in regulated industries (e.g., healthcare, finance) or when applications handle sensitive data.

● Adaptability and Extensibility: The modular design allows organizations to adapt the pipeline to evolving risks, new tools, and changing regulatory demands, thereby ensuring long-term relevance.

Development Velocity and Productivity Impact

While adding security modules could traditionally slow down development, automation helps maintain velocity. Static analysis and heuristic testing can run quickly—even in parallel—delivering near real-time feedback. Dynamic scanning and LLM-specific tests may add some time to build and deployment cycles, but given the value of preemptive detection, the trade-off is likely favorable.

Moreover, by reducing manual security audits, developer rework, and compliance-related overhead, overall productivity may improve. In many scenarios, early detection of vulnerabilities averts costly post-release patches, customer impact, reputational damage, and regulatory fines.

Trade-offs and Performance Overhead

● Resource Consumption: Heuristic-driven testing (especially via genetic algorithms) and dynamic scanning—especially in mobile-device emulation—can require substantial CPU, memory, or device resources. For large builds, this may slow down CI/CD throughput or require investment in dedicated infrastructure.

● False Positives/Negatives: Static analysis and heuristic tests may produce false positives (flagging benign code or behaviors) or false negatives (missing obscure runtime issues). Similarly, privacy leakage detection for LLMs may struggle to distinguish harmless generic responses from potentially sensitive leakage. Over-reliance on automation without human review increases risk of oversight.

● Maintenance Overhead: As dependencies, libraries, LLM models, and mobile platforms evolve, the pipeline modules must be updated, re-configured, or re-validated. Without careful governance, the pipeline might degrade in effectiveness over time.

● Adoption Barriers: Based on prior empirical studies (Jones, 2023; Lorona, 2023; Jammeh, 2020), organizations face cultural, educational, resource, and institutional barriers to DevSecOps adoption. Extending the pipeline with LLM-specific testing may increase complexity and resistance.

DISCUSSION

The proposed unified framework represents a deliberate attempt to converge two currently disjoint but increasingly overlapping domains: DevSecOps automation and security/privacy testing of LLM-embedded mobile applications. In doing so, we must consider not only technical design, but also organizational, cultural, and governance challenges.

Theoretical Implications

Our synthesis underscores that security in modern software systems can no longer be treated as an afterthought—or as a post-deployment responsibility. The integration of security modules at every stage of the pipeline—pre-commit, build-time, pre-deployment—embodies the principle of "shift-left," pushing

security earlier in the development lifecycle. This shift-left approach aligns with the foundational philosophy of DevSecOps (Jammeh, 2020; Hsu, 2019).

Moreover, by incorporating heuristic-driven testing (genetic algorithms), our framework expands the conceptual toolkit for vulnerability detection beyond traditional static and dynamic analysis. This is significant because many vulnerabilities—especially in complex, loosely-specified components like LLM libraries—may not manifest under deterministic tests, but rather under rare or context-sensitive inputs. Heuristic search brings probabilistic exploration to bear on such spaces, offering a promising strategy for uncovering hidden vulnerabilities (Thantharate & Anurag, 2023).

Finally, by including a dedicated LLM privacy/security validation module, we highlight the conceptual need for new kinds of security thinking in the age of AI-enhanced software. Traditional code-based analysis may not suffice; instead, we must incorporate data-flow, privacy leakage, behavioral robustness, and resource-exhaustion testing tailored to the unique properties of LLMs.

Practical and Organizational Challenges

While technically feasible, implementing the full unified framework in real-world organizations faces challenges:

● Cultural Resistance and Lack of Expertise: Many teams lack deep security expertise; as observed by (Jones, 2023) and (Lorona, 2023), security integration is often hindered by lack of training, low prioritization, or inadequate resource allocation. Introducing heuristic testing and LLM privacy testing may widen this expertise gap. Organizations may need to invest in training, hire security experts, or partner with external specialists.

● Infrastructure Costs: Running heuristic-driven and dynamic analyses (especially on device emulators or real devices) demands infrastructure—dedicated servers, mobile-device farms, or cloud-based testing environments. For smaller organizations or startups, such resource requirements may be prohibitive.

● False Sense of Security: Automated pipelines can create overconfidence: teams may assume that passing the pipeline equals security readiness. However, automation cannot guarantee detection of all vulnerabilities. Without human reviews, threat modeling, adversarial thinking, and penetration testing, residual risk remains. Compliance artifacts and test results may lull stakeholders into a false sense of safety.

● Governance and Compliance Complexity: Especially for regulated sectors, security and privacy compliance involve not just technical checks but legal, organizational, and policy-level considerations. The automated artifacts must map to regulatory frameworks (e.g., GDPR, HIPAA, PCI-DSS), which often require domain-specific interpretation, human judgment, and governance processes.

● Evolving Threat Landscape: As adversaries adopt more sophisticated tools—including AI-powered code analysis, adversarial inputs, and supply-chain attacks—the pipeline must adapt. Static analyzers, heuristics, and privacy testers will need regular updates. Otherwise, the pipeline will lag behind emergent threat vectors.

Recommendations for Adoption and Future Research

To translate the proposed framework into practice, we recommend the following steps:

● Pilot Implementation: Organizations interested in adoption should begin with a pilot project. For example, choose one mobile application that incorporates LLM features; implement the static analysis, heuristic-driven testing, dynamic scanning, and privacy testing modules; collect metrics (vulnerabilities found, false positives, build time overhead), and iterate.

● Cross-functional Teams and Training: Establish cross-functional DevSecOps teams combining developers, security engineers, privacy experts, and QA engineers. Provide training on heuristic testing, LLM privacy risks, and compliance documentation.

● Infrastructure Investment: Secure resources for automated testing infrastructure—device farms, sandbox environments, virtual machines—especially to support dynamic and LLM-specific testing.

● Human Review Gates and Risk Management: Complement automation with human oversight: manual security review gates, threat modeling sessions, adversarial-testing exercises, and periodic penetration tests. Automated results should feed into human decision-making.

● Continuous Monitoring and Feedback: After deployment, implement runtime monitoring (logs, anomaly detection, privacy audits) to capture potential post-release vulnerabilities or misuse. Feed this data back into the development pipeline to continuously improve security posture.

● Research Directions: Academia and industry should collaborate to evaluate the effectiveness of heuristic-driven vulnerability testing compared to traditional fuzzing; to develop standardized privacy leakage detection frameworks for LLMs; to design compliance artifact standards; and to study the human and organizational factors that promote or hinder adoption.

CONCLUSION

In a landscape where software development is accelerating, mobile applications increasingly embed large language models, and security threats continue to escalate, there is an urgent need for integrated, automated, and scalable security practices. This paper has synthesized the extant DevSecOps research on automated security testing—spanning static analysis, dynamic scanning, compliance artifact generation, and heuristic test generation—and proposed a unified framework that extends to LLM-enhanced mobile applications.

By combining static analysis, heuristic-driven testing, dynamic scanning, LLM-specific privacy/security validation, compliance artifact generation, and human feedback loops, the framework offers a comprehensive approach that balances security, privacy, compliance, and development velocity.

Implementing such a framework entails challenges—resource demands, cultural resistance, potential false confidence, and evolving threats—but through careful governance, training, and incremental adoption, organizations can significantly elevate their security posture. Future work should validate, refine, and empirically evaluate the framework, particularly in real-world settings with diverse applications, threat models, and regulatory environments.

Ultimately, embracing a holistic, automated, and adaptive security pipeline is no longer optional: it is essential for building resilient, trustworthy, and future-ready software in an increasingly complex and AI-enhanced digital ecosystem.

REFERENCES

1. Hsu, T. H. C. (2019). Practical security automation and testing: tools and techniques for automated security scanning and testing in DevSecOps. Packt Publishing Ltd.
2. Thantharate, P., & Anurag, T. (2023, September). GeneticSecOps: harnessing heuristic genetic algorithms for automated security testing and vulnerability detection in DevSecOps. In 2023, the 6th International Conference on Contemporary Computing and Informatics (IC3I) (Vol. 6, pp. 2271–2278). IEEE.
3. Marandi, M., Bertia, A., & Silas, S. (2023, July). Implementing and automating security scanning in a DevSecOps CI/CD pipeline. In 2023 World Conference on Communication and Computing (WCONF) (pp. 1–6). IEEE.
4. Jammeh, B. (2020). DevSecOps: Security expertise is a key to automated testing in the CI/CD pipeline. Bournemouth University.
5. Putra, A. M., & Kabetta, H. (2022, October). Implementation of DevSecOps by integrating static and dynamic security testing in CI/CD pipelines. In 2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM) (pp. 1–6). IEEE.
6. Abiola, O. B., & Olufemi, O. G. (2023). An enhanced CICD pipeline: A DevSecOps approach. International Journal of Computer Applications, 184(48), 8–13.
7. Lorona, N. (2023). Strategies Employed by Project Managers when Adopting Agile DevSecOps to Manage Software Development in the DoD (Doctoral dissertation, Colorado Technical University).

8. Jones, A. J. (2023). Quantitative Exploratory Investigation into the Barriers to Adopting DevSecOps Methodology for Security Operations Centers (Doctoral dissertation, Capitol Technology University).
9. Bitra, P., & Achanta, C. S. (2021). Development and Evaluation of an Artefact Model to Support Security Compliance for DevSecOps.
10. Security and Privacy Testing Automation for LLM-Enhanced Applications in Mobile Devices. (2025). International Journal of Networks and Security, 5(02), 30–41.