International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 3, May-June 2025||

DOI:10.15662/IJARCST.2025.0803001

Security in DevOps (DevSecOps): Integrating Security into the Development Pipeline

Raskhan

CMRIT, Kandlakoya, Medchal, India

ABSTRACT: DevSecOps represents the evolution of DevOps by embedding security practices directly into the development and delivery pipeline—shifting security from a gate at the end to a continuous, integrated concern across planning, coding, building, and deployment. This paper synthesizes the state of DevSecOps up to 2022, identifying key components: culture and collaboration across development, operations, and security; automation using Infrastructure as Code (IaC), Security as Code, and continuous security scanning; and governance via threat modeling, access control, and secrets management. Using insights from systematic reviews and empirical studies, we outline common challenges—such as cultural resistance, fragmented tooling, legacy systems, and lack of expertise—and mapped solutions like integrated security tools, developer-friendly security testing, and automated compliance pipelines. We propose a structured methodology for investigating DevSecOps practices: combining systematic literature review, empirical practitioner insights, and case study analyses to define workflows, assess benefits, and understand limitations. Key findings reveal that DevSecOps improves early vulnerability detection, maintains pipeline speed, and supports regulatory compliance when well implemented. We provide a workflow model incorporating threat modeling, security testing, prioritization, remediation, and monitoring within the CI/CD cycle. The advantages include faster feedback loops, reduced costs of fixing vulnerabilities, improved developer ownership, and auditability. However, drawbacks include increased complexity, training overhead, and possible tool fatigue. In conclusion, DevSecOps forms a pivotal step toward resilient software delivery, contingent on cultural alignment, tooling maturity, and leadership. Future research should explore longitudinal impact studies, scalable security tool integration, and explainability in security automation.

KEYWORDS: DevSecOps, security integration, CI/CD pipeline, shift-left security, threat modeling, automation, Infrastructure as Code, security as code, continuous security testing, security culture.

I. INTRODUCTION

DevOps accelerates software delivery by integrating development and operations, but it historically neglected comprehensive security integration—resulting in late discovery of vulnerabilities and operational risks. To address this, **DevSecOps** emerged, integrating security into every stage of the development pipeline. This shift-left approach ensures that security becomes a shared responsibility among development, operations, and security teams rather than a final gate.

Key drivers for DevSecOps include operational speed, regulatory compliance, and software trustworthiness. However, its adoption is challenged by cultural silos, tool fragmentation, skill gaps, and legacy infrastructures (turn0search0). Security must be reimagined as code, automated, and developer-friendly to ensure both security and velocity.

This paper lays the foundation for systematically understanding DevSecOps, drawing on empirical reviews such as the systematic study by Rajapakse et al., which analyzes the challenges and solutions practitioners encounter when adopting DevSecOps across people, practices, tools, and infrastructure domains (turn0academia13). Empirical insights from security tool integration further highlight evolving tool ecosystems and persistent friction in CI/CD workflows (turn0academia15).

Through integrated methodologies—literature reviews, practitioner studies, and case examples—this work explores how organizations can effectively embed security without undermining agility. The introduction sets up DevSecOps as both a cultural and technical transformation geared toward continuous, automated, and collaborative security in the software lifecycle.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 3, May-June 2025||

DOI:10.15662/IJARCST.2025.0803001

II. LITERATURE REVIEW

The academic and practitioner literature up to 2022 provides a multifaceted view of DevSecOps:

1. **Challenges and Solutions (Systematic Review)**
2. Rajapakse et al. conducted a systematic literature review of 54 studies on DevSecOps adoption. They identified 21 challenges—most related to tooling and automation demands—and mapped 31 corresponding solutions. Key themes include the need for shift-left practices and continuous security assessment, supported by developer-centric testing tools (turn0academia13).
3. **Security Tool Integration (Empirical Analysis)**
4. A qualitative study of DevOps practitioners revealed that traditional security tools are often incompatible with modern CI/CD workflows. Practitioners recommend new tool types designed for integration, providing guidelines for embedding security tools with minimal disruption (turn0academia15).
5. **Security Automation in CI Pipelines (Case Study of OSS)**
6. Angermeir et al.'s case study on enterprise OSS projects showed that only 6.83% of projects integrated automated security into their CI pipelines—revealing a notable gap between recognizing security importance and its automation (turn0academia16).
7. **Standards Compliance in Industrial Contexts**
8. A case study integrating the IEC 62443-4-1 security standard into DevOps pipelines for industrial control systems demonstrated that security compliance can be automated while retaining agility, balancing regulatory needs and fast delivery (turn0academia17).

Overall, the literature emphasizes that while DevSecOps offers significant benefits, its realization is hindered by cultural, technical, and organizational barriers. Tool support, automation, and governance frameworks emerge as critical enablers, particularly when partnered with cultural and leadership commitment.

III. RESEARCH METHODOLOGY

This paper proposes a mixed-method research methodology to explore DevSecOps implementation holistically:

1. **Systematic Literature Review (SLR)**
2. Following Rajapakse et al., we perform an updated SLR restricted to pre-2022 literature, focusing on challenges, solutions, and best practices in DevSecOps. Search terms include “DevSecOps”, “security in CI/CD”, and “shift-left security”.
3. **Empirical Data Collection**
4. Analyze practitioner perspectives through secondary data from webinars and practitioner reports (as done in the tool-integration study [turn0academia15]). We employ thematic analysis to synthesize common friction points and effective practices across people, tools, and processes.
5. **Case Study Analysis**
6. Incorporate detailed case studies from literature: OSS projects’ low adoption of security automation ([turn0academia16]) and industrial integration of security standards in pipelines ([turn0academia17]), to illustrate practical application and impact.
7. **Workflow Construction**
8. Based on SLR and case insights, define a formal DevSecOps workflow model, detailing stages, automation tools, stakeholder responsibilities, and feedback loops.
9. **Comparative Framework**
10. Contrast traditional DevOps pipelines with DevSecOps-enhanced versions across key metrics: vulnerability detection timing, deployment cycle time, and compliance. Qualitative assessment is informed by case data and practitioner feedback.
11. **Gap Analysis & Future Needs**
12. Identify remaining gaps—such as tool limitations, scalability concerns, or cultural adoption obstacles—and propose areas for future investigation.

This methodology integrates theoretical, empirical, and contextual insights to construct a practical, evidence-based understanding of DevSecOps adoption.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 3, May-June 2025||

DOI:10.15662/IJARCST.2025.0803001

IV. KEY FINDINGS

Based on synthesized literature and case data, the following key insights emerged:

1. **Cultural Shifts Are Essential**
2. DevSecOps requires breaking down silos between development, security, and operations. Leadership support and training are crucial; security must be everyone's responsibility (turn0search7).
3. **Shift-Left Enables Early Detection**
4. Integrating SAST, DAST, and IAST early in the development lifecycle reduces cost and severity of vulnerabilities (turn0search5; turn0search1; turn0search18).
5. **Tooling Must Be CI/CD Friendly**
6. Practitioners report that legacy security tools often disrupt pipeline speed. New automation-friendly tools are better suited for DevSecOps workflows (turn0academia15).
7. **Security Standards Can Be Automated**
8. Case studies integrating IEC 62443-4-1 into CI pipelines show that regulatory security controls can be implemented without sacrificing agility (turn0academia17).
9. **Low Automation Adoption Remains an Issue**
10. Even in OSS environments, security automation is rare (~7%), underscoring the gap between security awareness and actual practice (turn0academia16).
11. **Systematic Best Practices Exist**
12. Effective DevSecOps pipelines include threat modeling, security testing, prioritization, remediation, and monitoring. Infrastructure as Code and Security as Code are core enabling techniques.

Altogether, DevSecOps presents measurable benefits in speed, detection efficiency, and compliance when supported by the right culture, tools, and processes.

V. WORKFLOW

An effective DevSecOps pipeline integrates security seamlessly into the CI/CD loop. The workflow is as follows:

1. **Threat Modeling (Planning Phase)**
2. DevSecOps begins with threat modeling to identify and assess potential security risks. Teams define attack surfaces, threat actors, and mitigation strategies before development begins (turn0search1).
3. **Secure Coding & Pre-commit Checks (Coding Phase)**
4. Developers adhere to secure coding standards and enforce pre-commit hooks—for example, static analysis or secret scanning integrated directly into IDEs—providing immediate feedback (turn0search11; turn0search8).
5. **Continuous Security Testing (CI Stage)**
6. Automated SAST, DAST, and IaC policy checks are embedded into the CI pipeline, scanning code, dependencies, and configurations for vulnerabilities .
7. **Analysis and Prioritization**
8. Security findings are aggregated and prioritized based on risk, impact, and exploitability, balancing remediation needs with delivery timelines (turn0search1).
9. **Remediation and Security as Code**
10. Developers fix critical issues promptly. Security policies and configurations are codified (Security-as-Code, Infrastructure-as-Code) for consistency and auditability (turn0search5).
11. **Deployment with Compliance Gates**
12. Pipelines enforce security policies at deployment gates, potentially integrating industry standards like IEC 62443 across environments (turn0academia17).
13. **Monitoring & Feedback (Post-deployment)**
14. Runtime monitoring, pentesting, and bug bounty programs provide continuous visibility into production vulnerabilities (turn0search1). Feedback informs future iterations.
15. **Continuous Improvement Loop**
16. Metrics on detection latency, false positives, and pipeline efficiency are used to refine threat models, tooling, and governance.

This workflow aligns with the DevSecOps “infinity loop,” embedding security throughout and minimizing friction between speed and safety.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 3, May-June 2025||

DOI:10.15662/IJARCST.2025.0803001

VI. ADVANTAGES & DISADVANTAGES

Advantages

- **Early Detection**: Shift-left practices reduce costs and risks associated with late vulnerability discovery.
- **Automation & Consistency**: Security-as-Code and IaC ensure reproducibility and compliance.
- **Cultural Ownership**: Shared responsibility promotes broader security awareness across teams.
- **Compliance Readiness**: Security checks and standards can be automated into pipelines.
- **Minimal Pipeline Disruption**: Modern tools compatible with CI/CD preserve velocity.

Disadvantages

- **Tooling Complexity**: Integrating and maintaining security tools may add pipeline complexity.
- **Need for Training**: Teams must learn secure coding and new tools, which may slow adoption.
- **Alert Fatigue**: Excessive or poorly prioritized findings can overwhelm teams.
- **Legacy System Challenges**: Older systems may not support modern integration approaches.
- **Cultural Resistance**: Organizational inertia and siloed mindsets can hinder change (turn0search0).

VII. RESULTS AND DISCUSSION

Empirical and case-based evidence indicates that DevSecOps enhances both security posture and delivery speed. Shift-left security practices, like SAST/DAST, catch vulnerabilities early, reducing remediation costs and cycle impact (turn0search5; turn0search1). Automating security as part of code and infrastructure ensures traceability and reduces configuration drift.

In environments where security standards were automated (e.g., IEC 62443), organizations achieved both compliance and agility—contradicting concerns that security necessarily slows delivery (turn0academia17). Low automation uptake in OSS projects suggests potential for improvement, emphasizing the need for tooling and awareness (turn0academia16).

Risk remains when tool integration is poorly executed. Practitioner feedback highlights potential breaks in pipeline flow if tools are not CI-native. Smooth integration requires tools designed for automation and developer visibility (turn0academia15). Cultural change remains a non-trivial obstacle, emphasizing leadership as a foundational enabler (turn0search7).

In summary, DevSecOps achieves the dual goals of speed and security when supported by automated tools, integrated policies, and a culture of shared responsibility. However, implementation requires deliberate effort, tooling maturity, and organizational alignment.

VIII. CONCLUSION

This study synthesizes the evolution and key aspects of **DevSecOps**, demonstrating that integrating security into the development pipeline is both feasible and beneficial. Key findings highlight that DevSecOps:

- Enhances early vulnerability detection and reduces remediation costs through shift-left security.
- Stabilizes delivery pipelines through automation (SAST, DAST, security-as-code, IaC).
- Supports compliance and resilience via automated standards integration (e.g., IEC 62443).
- Requires cultural transformation, leadership, and training to overcome silos and adoption resistance.

The general workflow—from threat modeling through monitoring—embeds security into every stage of delivery, aligning security with speed. While complexities remain—integration, tool fatigue, legacy constraints—careful implementation can mitigate these.

Ultimately, DevSecOps represents a vital evolution in modern software engineering, aligning security with agile delivery while maintaining trust, compliance, and efficiency.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 3, May-June 2025||

DOI:10.15662/IJARCST.2025.0803001

IX. FUTURE WORK

Moving forward, these areas merit further exploration:

1. **Longitudinal Studies**
2. Investigate DevSecOps' impact over time on security incidents, delivery metrics, and team behavior to quantify long-term benefits and ROI—addressing current limitations on sample size and temporal insight (turn0search7).
3. **Scalability in Large Organizations**
4. Compare adoption patterns across small, medium, and large enterprises to tailor best practices by scale (turn0search7).
5. **Tooling Evolution and AI Integration**
6. Assess the next generation of developer-friendly security tools and AI-driven risk prioritization to enhance automation and reduce alert fatigue (turn0academia15).
7. **Explainable Security Automation**
8. Incorporate explainability into automated security findings to help developers understand and trust remediation guidance.
9. **Legacy System Integration**
10. Develop strategies and hybrid approaches to introduce DevSecOps practices into legacy systems where CI/CD may not be fully established.
11. **Cross-domain Standardization**
12. Establish frameworks and templates for automating diverse industry standards (e.g., ISO, NIST, IEC) in pipelines.
13. **Security Culture Interventions**
14. Evaluate interventions—training programs, security champions, incentive models—for cultivating broad security ownership.

REFERENCES

1. Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2021). Challenges and solutions when adopting DevSecOps: A systematic review. *arXiv preprint* (turn0academia13).
2. Rajapakse, R. N., Zahedi, M., & Babar, M. A. (2021). An empirical analysis of practitioners' perspectives on security tool integration into DevOps. *arXiv preprint* (turn0academia15).
3. Angermeir, F., Voggenreiter, M., Moyón, F., & Mendez, D. (2021). Enterprise-driven open source software: A case study on security automation. *arXiv preprint* (turn0academia16).
4. Moyón Constante, F., Soares, R., Pinto-Albuquerque, M., Méndez, D., & Beckers, K. (2021). Integration of security standards in DevOps pipelines: An industry case study. *arXiv preprint* (turn0academia17).
5. Codefresh. DevSecOps Pipeline: Steps, Challenges, and 5 Critical Best Practices -2022 content). (turn0search0).
6. HackerOne. 5 Security Stages of the DevSecOps Pipeline. (turn0search1).
7. Quest Technology Management. Top DevSecOps Best Practices to Secure Your Development Pipeline -. (turn0search2).
8. Synopsys. Building Your DevSecOps Pipeline: 5 Essential Activities. (turn0search11).
9. ResearchGate. Integrating Security Into the DevOps Process (DevSecOps) -. (turn0search4).
10. Wikipedia. Static Application Security Testing (SAST) (turn0search18).
11. Wikipedia. Microsoft Security Development Lifecycle (SDL). (turn0search19).
12. IET Software. Revisiting security in the era of DevOps: An evidence-based inquiry into DevSecOps industry.