

# DevSecOps: A Comprehensive Survey of Secure Software Development and Deployment Practices

**Dr. Prashant Kumar Shrivastava**

PhD CSE and Associate Professor

Sanjeev Agrawal Global Educational (SAGE) University, Bhopal

[prashant.s@sageuniversity.edu.in](mailto:prashant.s@sageuniversity.edu.in)

**Abstract:** *DevSecOps brings together development, security, and operations approaches to incorporate security across the software development lifecycle, allowing organizations to discover and address vulnerabilities earlier while ensuring quick and dependable software delivery. This survey reviews the foundations, practices, technologies, challenges, and recent research associated with secure software development and deployment. The Secure SDLC is dissected, from security requirements and threat modeling to secure coding, testing, deployment, and continuous monitoring. The study further discusses shift-left security, continuous security testing, automated vulnerability management, Security as Code, CI/CD integration, and compliance automation. Key automated security technologies, including SAST, DAST, Software Composition Analysis, and Infrastructure-as-Code security scanning, are reviewed. The survey also highlights the growing role of AI and ML in vulnerability identification, threat intelligence, security analysis, and adaptive security automation. Recent literature is compared to identify major findings, limitations, and research directions. Overall, DevSecOps provides a proactive and collaborative approach for improving software security, resilience, scalability, compliance, and deployment efficiency while addressing organizational, legacy-system, toolchain, and performance challenges across increasingly complex cloud-native and distributed environments today.*

**Keywords:** DevSecOps, Secure Software Development, Security Automation, CI/CD, Artificial Intelligence.

## I. INTRODUCTION

Traditional software development models, including waterfall, V-model, and spiral approaches, provide structured processes for software planning, development, testing, and delivery. However, their limited flexibility can make it difficult to accommodate rapidly changing requirements, frequent releases, and evolving security threats [1]. Cloud computing, microservices, containers, CI/CD, agile development, and other modern software development approaches have sped up the software development process but have also increased the necessity of security processes that run constantly throughout the product lifecycle [2][3].

Several phases of the software development lifecycle (SDLC), such as requirements analysis, design, coding, testing, deployment, and maintenance, are integrated with security concerns in secure software development. Vulnerabilities may be found and fixed before they have an impact on applications by using techniques like threat modeling, secure coding, vulnerability assessment, and security testing. However, treating security as a separate or late-stage activity can introduce delays and leave vulnerabilities unresolved [4]. This constraint has aided the rise of DevSecOps, which combines security with development and operations to enable safe and continuous software delivery.

DevSecOps represents a security-focused extension of DevOps that embeds security responsibilities, practices, and controls throughout the development and deployment pipeline. It promotes collaboration among development, security, and operations teams while incorporating security activities into CI/CD workflows [5]. The shift-left approach



introduces security earlier in the development process, enabling vulnerabilities to be identified and addressed before deployment. Automation further supports continuous security through automated testing, vulnerability scanning, compliance validation, and monitoring, thereby reducing manual effort while maintaining development efficiency [6]. The growing use of cloud-based applications, distributed systems, open-source components, containers, and third-party services has further expanded the software attack surface [7]. Consequently, modern DevSecOps environments require practices such as static and dynamic application security testing (SAST/DAST), software composition analysis (SCA), infrastructure-as-code (IaC) security, container security, secrets management, and continuous monitoring. These practices help organizations maintain security across both software development and deployment processes [8][9]. Recent advances in artificial intelligence (AI) have also created opportunities for automated vulnerability detection, code analysis, threat identification, and security testing. However, AI adoption introduces challenges related to reliability, explainability, privacy, and model security. Therefore, understanding how security, automation, and emerging technologies integrate within DevSecOps is essential for reliable and resilient software delivery. This survey comprehensively examines DevSecOps principles, secure development and deployment practices, tools, automation approaches, challenges, and emerging trends.

This paper is divided into five sections: Section II introduces Secure Software Development in DevSecOps. Section III explores Security Automation and Technologies in DevSecOps. Section IV reviews recent literature, and Section V concludes the paper while outlining potential future research directions.

## **II. SECURE SOFTWARE DEVELOPMENT IN DEVSECOPS**

DevSecOps, which is short for development, security and operations, is an application development practice that automates the integration of security practices at every phase of the software development lifecycle, from initial design through integration, testing, delivery and deployment [10]. DevSecOps represents a natural and necessary evolution in the way development organizations approach security. In the past, security was ‘tacked on’ to software at the end of the development cycle, almost as an afterthought. A separate security team applied these measures, and then a separate quality assurance (QA) team tested them.

### **A. Secure Software Development Life Cycle**

The goal of implementing a secure SDLC is to enhance software security, dependability, and early vulnerability detection. As shown in Fig. 1. Secure SDLC includes planning and requirements, secure design, implementation, testing, deployment, and maintenance, with security activities incorporated throughout each phase.

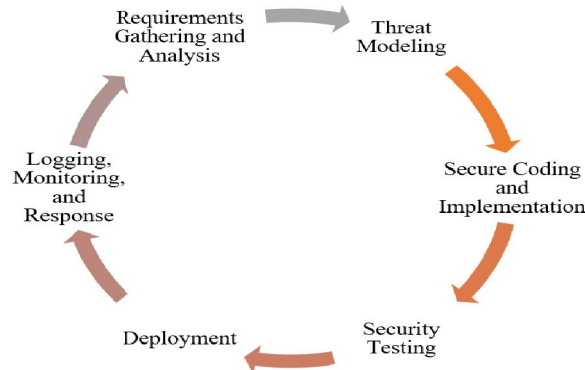


**Fig. 1. Secure Software Development Life Cycle**

- **Requirements Gathering and Analysis:** This is the first phase, and it involves security requirements being identified alongside functional requirements. Security experts collaborate with stakeholders to determine potential threats and establish security objectives.



- **Threat Modeling:** This is the second phase, and it involves Threat Modelling. Threat modeling is a methodical strategy for finding, assessing, and reducing security threats in software applications or systems. It aids in identifying possible risks and vulnerabilities in system or application software.
- **Secure Coding and Implementation:** This is the third step, in which developers write secure code using secure coding rules and best practices [11]. They use secure APIs, input validation techniques, and encryption to protect against common vulnerabilities like SQL Injection and Cross-Site Scripting (XSS).
- **Security Testing:** This is the fourth phase, and it involves Develops engineers and security engineers performing security testing at various levels, including Static Analysis, Dynamic Analysis, and penetration testing.
- **Deployment:** This is the fifth phase, and security measures continue during deployment. Secure configurations, strong authentication, and encryption of data in transit and at rest are to be ensured.
- **Logging, Monitoring, and Response:** This is the sixth phase, and once the software is deployed, continuous logging and monitoring are essential. Security logs are analyzed, and any suspicious activities are investigated promptly. The implementation of incident response strategies guarantees a prompt and efficient reaction to any security issues that may arise, including breaches.

#### **B. Integrating Security into the DevSecOps Lifecycle**

Security is the main issue in the DevSecOps Lifecycle. To resolve this issue, integration of security happened some of the integrated security listed below:

- **Shift-left approach to security:** The shift-left approach to security focuses on incorporating security practices and testing earlier in the software development lifecycle, rather than treating it as a separate, late-stage activity. Organizations may find and fix vulnerabilities more quickly and cheaply by addressing security issues throughout the planning, design, and coding phases.
- **Continuous security testing and monitoring:** Implementing automated security scans and vulnerability assessments as part of the CI/CD pipeline helps to identify and address security issues early on [12]. Continuous monitoring of applications, infrastructure, and cloud resources for security threats, using tools like SIEM (Security Information and Event Management) and CSPM (Cloud Security Posture Management), enables rapid detection and response to security incidents.
- **Automated security scans and vulnerability management:** Software engineers may automate security scans, vulnerability discovery, and repair by integrating security tools and scripts into the CI/CD pipeline. Automating the process of identifying, prioritizing, and addressing vulnerabilities reduces manual overhead and ensures security issues are addressed in a timely manner.

#### **C. The stages of the SDLC**

The final set of SDLC stages and their brief explanation are as follows:

- **Project inception, or planning phase,** provides a high-level overview of the project goals and requirements, along with preliminary activities for software development.
- **Analysis and requirements phase** involves creating and maintaining software requirements.
- **Architectural and detailed design phase** “identifies the major components of the system and the communication between these components” [13].
- **Implementation phase** involves writing a program based on the requirements.
- **Verification and testing phase** ensures that the software meets the requirements and fulfills the customer’s expectations.
- **Release and maintenance phase** includes activities related to release preparation, deployment, and post-production maintenance.
- **Disposal phase** involves activities to retire the software.



#### **D. Challenges In DevSecOps**

Adopting DevSecOps in large organizations presents several significant challenges. Organizational resistance is often the most prominent barrier, as teams accustomed to traditional, siloed workflows may resist changes to processes and culture.

- **Organizational Resistance and Cultural Barriers:** One of the most significant barriers to adopting DevSecOps in large organizations is organizational resistance. Transitioning from traditional silos of development, operations, and security to a unified DevSecOps model requires a significant cultural shift.
- **Complexity of Existing Legacy Systems:** Large organizations often rely on legacy systems—outdated software or hardware platforms that are integral to business operations but difficult to modernize.
- **Toolchain Integration Issues:** DevSecOps relies heavily on an integrated toolchain for automation, security testing, and monitoring [14]. However, toolchain integration is a common challenge in large organizations with diverse technology stacks.
- **Scalability and Performance Constraints:** As organizations scale, the performance and scalability of their DevSecOps processes become critical challenges. Large enterprises often work with complex, multi-cloud, and hybrid IT environments that require a robust security infrastructure to support thousands of applications, services, and users.

#### **E. Key Benefits of DevSecOps**

Integrating security into the process entails thinking about it from the beginning of a feature's thought process all the way through to the product's deployment.

- **Iterative and Incremental Development:** This practice involves a cyclical approach to breaking software development into smaller, more manageable steps. Continuous integration/continuous deployment (CI/CD) practices provide the automation necessary to ensure quality, security, and functionality.
- **Continuous Feedback:** Feedback should be collected throughout every step of the lifecycle to allow for expert validation and general observability [15]. Every tool used throughout a DevSecOps pipeline creates some output that can be utilized for feedback.
- **Metrics and Measurement:** Metrics are used to evaluate feedback and determine how well the organization is performing on measures, such as productivity and quality.
- **Transparency and Traceability Across the Lifecycle:** It builds the trust needed among the Dev, Sec, and Ops teams to make the process work, and traceability enables the digital trail of the products utilized to build, deploy, and maintain software.

#### **F. Software Development Approaches**

Development approaches allow organizations to boost customer satisfaction, increase development results, and produce secure software of high quality [16]. There are a variety of strategies that organizations and development teams may use to steer the software development process [17]. Table I lists a few of the most popular methods.

**TABLE I. COMMON SOFTWARE DEVELOPMENT APPROACHES**

| <b>Approach</b> | <b>Particulars</b>                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Agile           | Scrum and Kanban are two examples of agile approaches that place an emphasis on iterative development, collaboration, and adaptability [18].<br>Teams move in short bursts, termed sprints, and produce usable software increments at the conclusion of each cycle.<br>Continuous cooperation across cross-functional teams and stakeholders leads to evolving requirements and solutions. |
| Waterfall       | Waterfall is a method for developing software that is linear and sequential [19].<br>The process has established steps, such as requirements gathering, design, implementation, testing, deployment, and maintenance. Each phase has particular objectives and deliverables, and progress is                                                                                               |



|                          |                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          | made incrementally from one to the next.                                                                                                                                                                                                                                                                                                                                                      |
| Extreme Programming (XP) | Extreme Programming is a software development process that prioritizes collaboration, open communication, and top-notch software [20].<br>It encourages customer interaction, pair programming, regular releases, continuous testing, and collaborative code ownership.<br>XP prioritizes ease of use, adaptation to changing requirements, and flexibility.                                  |
| DevOps                   | DevOps is a method that brings together the technical and cultural aspects of software development and operations in a way that promotes teamwork and communication [21].<br>Its goal is to automate and simplify software distribution by deploying it often and reliably.<br>Infrastructure as code, automated testing, continuous integration, and continuous delivery are key approaches. |
| Lean                     | Lean prioritizes cutting down on waste, increasing efficiency, and providing ongoing value to customers [22].<br>It places a strong emphasis on workflow optimization, process simplification, and the removal of non-value-adding tasks.<br>Software development may be made more efficient and rapid by combining Lean concepts with Agile approaches [23].                                 |

### III. SECURITY AUTOMATION AND TECHNOLOGIES IN DEVSECOPS

Automation is essential to DevSecOps because it makes it possible to test and validate security measures quickly and consistently throughout the development process. By identifying vulnerabilities early in the development lifecycle, automated security tests enable teams to quickly fix them and lower the likelihood of security breaches. Various tools and technologies are available to support automated security checks in DevSecOps pipelines [24]. These include container security tools that look for vulnerabilities in container images, DAST tools that mimic actual assaults, and static code analysis tools that scan code for possible security flaws.

#### A. Evolution of DevSecOps and Security Practices

Security traditionally operated in isolation, addressing concerns only after software was developed and deployed. This reactive approach led to costly vulnerabilities and incidents [25]. DevSecOps aims to 'shift security left,' incorporating security measures during development rather than post-deployment. The evolution of this practice includes:

- **Security as Code (SaC):** Automating security policies and practices within the development pipeline. By treating security as code, teams can implement version control, peer reviews, and automated testing for security measures, ensuring consistent and reliable enforcement.
- **Continuous Security Monitoring:** Maintaining a secure application and infrastructure with the use of development tools that reveal vulnerabilities in real time. Continuous monitoring is essential for detecting and swiftly addressing security concerns. It helps to avoid vulnerabilities from being exploited in production.
- **Integration with CI/CD Pipelines:** Modern DevOps practices involve integrating security tools into CI/CD pipelines. This integration lowers the possibility of security breaches by ensuring that each code update is automatically checked for vulnerabilities prior to deployment.
- **Adaptive Security:** With the use of ML and AI, DevSecOps can adapt to evolving threats by learning from past incidents. Adaptive security measures enable dynamic policy adjustments based on the changing threat landscape, making security more responsive and proactive.

#### B. Tools for Automated Security Testing

In DevSecOps, automation plays a key role in preserving consistency and speed while guaranteeing application security. Developers may find vulnerabilities at any point in the development lifecycle, from developing code to deploying it, with the aid of the many automated security testing tools available [26]. These tools can be categorized



based on the type of testing they perform, such as SAST, DAST, Software Composition Analysis (SCA), and Infrastructure as Code (IaC) security scanning.

- **Static Application Security Testing (SAST) Tools:** SAST tools do static analysis on binaries, bytecode, or source code to identify vulnerabilities before an application is ever built, all without executing the code. These tools examine code for issues such as buffer overflows, SQL injection, cross-site scripting (XSS), and more. They provide immediate feedback to developers, allowing them to address security flaws before they propagate to later stages of the development pipeline.
- **Dynamic Application Security Testing (DAST) Tools:** Unlike SAST, DAST tools test an application in its running state, simulating real-world attacks to identify runtime vulnerabilities that may not be visible in the source code. DAST tools perform penetration testing by interacting with the application's user interface (UI) or APIs to identify issues such as broken authentication, security misconfigurations, and data exposure.
- **Software Composition Analysis (SCA) Tools:** SCA tools focus on identifying and managing the security risks associated with open-source components and third-party libraries used in application development. These utilities will check for out-of-date dependency versions, known vulnerabilities, and licensing concerns automatically.
- **Infrastructure as Code (IaC) Security Tools:** IaC has become increasingly popular for managing cloud infrastructure, but misconfigurations in IaC scripts can lead to severe security vulnerabilities. IaC security tools help automate the detection of such vulnerabilities by scanning infrastructure code for misconfigurations, insecure settings, and potential security risks.

### **C. Automated Security and Compliance Framework Implementation**

Implementing the proposed framework requires integrating various tools and processes into the CI/CD pipeline [27].

The following steps outline the implementation process:

- **Policy Definition:** Define compliance policies as code using OPA or HashiCorp Sentinel. These policies should cover regulatory requirements relevant to the organization.
- **Tool Integration:** Integrate security and privacy testing tools such as OWASP ZAP, SonarQube, and Privacy Badger into the CI/CD pipeline.
- **IaC Configuration:** Configure Terraform or AWS Config to enforce compliance policies on infrastructure resources.
- **Pipeline Automation:** Compliance checks should be carried out automatically as part of the CI/CD process. This entails setting up the pipeline to automatically execute the specified rules and tests whenever infrastructure is provided or code is committed.
- **Monitoring and Reporting:** Implement monitoring and reporting mechanisms to track compliance status and identify potential issues. This includes configuring alerts to notify relevant teams of any compliance violations.

### **D. AI-Based Security Automation**

The goal of approach is to improve the security analysis process within the DevSecOps pipeline. Instead of manually extracting useful information from the scanning reports generated by SAST and DAST, leverage the power of AI to automatically analyze these reports and generate a comprehensive security analysis report [28]. This enables development teams to quickly address potential security defects. As SAST and DAST generate two different types of scanning results by their inherent nature, firstly preprocess them to a JSON summary for later usage by AI application.

### **E. Role of AI in Identifying Vulnerabilities**

The process of identifying, analyzing, and prioritizing vulnerabilities is radicalized by artificial intelligence. AI-based technologies may minimize human participation in normal security tasks, spot anomalies, and make judgments about risk based on minor signs in the code by analyzing previous data and recognized patterns of attacks.

- **Machine Learning Algorithms:** Machine learning (ML) algorithms may be taught to spot suspicious patterns that point to vulnerable code, incorrect setup, or unusual activity by using large datasets of known



vulnerabilities. Using probabilistic-aided static analysis, such models may be used to discover vulnerable code sections [29]. Additionally, it details the enterprise-wide integration of supervised learning algorithms into AI security solutions for the purpose of analyzing commit histories, identifying code patterns linked to vulnerabilities, and detecting possible hazards in real-time.

- **Natural Language Processing for Threat Intelligence:** NLP has opened up new avenues for threat intelligence mining by automating the analysis of unstructured languages such as CVE reports, hacker forums, and dark web activities. NLP-powered technologies may add context-based knowledge to information enrichment databases by offering signs of compromise, attack tactics, and vulnerabilities.

#### **IV. LITERATURE REVIEW**

This section highlights related studies on DevSecOps-based security practices, focusing on automated security testing, secure CI/CD pipelines, vulnerability management, continuous monitoring, compliance, cloud-native security, and risk management to improve the security, reliability, scalability, and efficiency of ModernSoftwareDevelopment and deployment.

Dheeraj Velaga et. al. (2026) a software engineering framework powered by DevSecOps for secure and fast deployment of ModernSoftware applications in cloud-native and distributed environments. It integrates automated security measures, including SAST/DAST, container vulnerability testing, IaC security testing, and continuous compliance testing into the CI/CD pipeline. Experimental analysis shows that the proposed framework outperforms traditional DevOps workflows in deployment speed, vulnerability detection rate, and system reliability [30].

Sai Niranjan, V. Rithik Sakthi B and J. Sathiamoorthy (2026) presents DevOps-powered security integration platform, or to be more specific, DevSecOps, which is the composite of SecurityControl and VulnerabilityScanning alongside confirmation that CI/CD pipelines meet the standards. This may enable development teams to identify and address vulnerabilities earlier in the software life cycle, mitigate risk exposure and release software more quickly by shifting security to the left. The architecture contained in the proposed scheme presents an architecture that is tool-independent and converts the benefits, both performance and security, to a real-life deployment scenario [31].

S. D. L. V. Dasanayake, Janaka Senanayake and W. M. J. I. Wijayanayake (2025) discuss the benefits of embedding automated security testing and continuous monitoring to enhance resilience against evolving threats. Tailored developer training addresses knowledge gaps specific to trading platforms, ensuring regulatory compliance and safeguarding sensitive financial data. By accelerating deployment timelines while strengthening security and compliance, this study demonstrates the critical role of a DevSecOps model in creating scalable, secure, and resilient trading applications [32]. Mengxia Chen, Bing Liang and Xiaofei Lu (2024) presented cloud internet integration in China Telecom by highlighting the DevSecOps platform. The overall architecture of the self-developed Cloud platform and the security center was introduced and discussed. The extensive use of the self-developed DevSecOps platform not only demonstrates successful use of the platform but also underscores the high credibility of the security system. This work provides a paradigm for the tailoring and development of a large-scale commercial DevSecOps platform [33].

Mohammad A. Aljohani and Sultan S. Alqahtani (2023) discuss current challenges in securing DevOps applications, including the lack of automated software security testing tools, insufficient integration of security tools, a lack of security knowledge among developers, and false-positive results from many vulnerability scanners. As a result, the DevSecOps paradigm requires the creation of a centralized framework for automating software security analysis, which will serve as a bridge between the software application's CI/CD pipelines and the application security services. It has shown the framework's high-level architecture, and one case study is presented to illustrate the applicability of the proposed approach [34].

Xhesika Ramaj (2022) presents a Ph.D. research plan that focuses on addressing existing problems in the risk management of critical infrastructure through a novel DevSecOps-enabled framework. Critical infrastructure is complex physical and cyber-based systems that form the lifeline of modern society, and their reliable and secure



operation is of paramount importance to national security and economic vitality. Therefore, this paper proposes DevSecOps technology for managing risk throughout the entire development life cycle of such systems [35].

Table II provides a summary of key studies on recent DevSecOps studies based on their focus, methodologies, findings, limitations, and future directions, highlighting the role of automated security testing, CI/CD integration, continuous monitoring, Vulnerability Management, and secure software deployment.

**TABLE II. COMPARATIVE ANALYSIS OF RECENT DEVSECOPS APPROACHES FOR SECURE SOFTWARE DEVELOPMENT AND DEPLOYMENT**

| Reference                                                                 | Study Focus                                                                              | Approach / Methodology                                                                                                                                                | Key Findings                                                                                                                                | Challenges / Limitations                                                                                                    | Future Directions                                                                                                                     |
|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| D. Velaga et al. (2026)                                                   | Secure and rapid deployment of cloud-native and distributed applications using DevSecOps | Proposed a DevSecOps framework integrating SAST, DAST, container vulnerability testing, IaC security testing, and continuous compliance testing into CI/CD pipelines. | The framework outperformed traditional DevOps in deployment speed, vulnerability detection, and system reliability.                         | Requires integration of multiple security mechanisms and adaptation to complex cloud-native environments.                   | Develop more automated, intelligent, and scalable security controls for distributed and cloud-native applications.                    |
| S. Niranjana V., R. Sakthi B. and J. Sathiamoorthy (2026)                 | Integration of security controls and vulnerability scanning into DevOps pipelines        | Developed a tool-independent DevSecOps architecture integrating security controls, vulnerability scanning, and CI/CD compliance validation.                           | Enables early vulnerability identification, reduces security risks, supports shift-left security, and improves software release efficiency. | Tool-independent integration may require customization for different development environments and security infrastructures. | Improve interoperability, automation, and adaptability across diverse CI/CD platforms and tools.                                      |
| S. D. L. V. Dasanayake, J. Senanayake and W. M. J. I. Wijayanayake (2025) | DevSecOps for secure and resilient trading applications                                  | Integrated automated security testing, continuous monitoring, developer training, and compliance practices for financial trading platforms.                           | Improved deployment speed, security, regulatory compliance, and resilience while protecting sensitive financial data.                       | Focused on trading applications; developer knowledge gaps and strict regulatory requirements remain challenges.             | Extend DevSecOps practices to other financial and mission-critical applications and develop stronger automated compliance mechanisms. |
| M. Chen, B. Liang and X.                                                  | Large-scale DevSecOps                                                                    | Presented a self-developed                                                                                                                                            | Demonstrated successful large-                                                                                                              | Large-scale commercial                                                                                                      | Enhance platform scalability,                                                                                                         |



|                                           |                                                                |                                                                                                                                                          |                                                                                                                                                |                                                                                                               |                                                                                                                                                |
|-------------------------------------------|----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Lu (2024)                                 | implementation in a cloud environment                          | DevSecOps platform, including cloud platform architecture and an integrated security center, and evaluated its practical deployment.                     | scale implementation and improved the effectiveness and credibility of the security system.                                                    | implementation involves complexity, customization, and operational costs.                                     | automation, intelligent security analysis, and applicability to other large-scale enterprises.                                                 |
| M. A. Aljohani and S. S. Alqahtani (2023) | Automated software security analysis within DevSecOps          | Proposed a unified security-analysis framework positioned between CI and CD pipelines and application security services; evaluated through a case study. | Addressed inadequate security testing, poor tool integration, limited developer security knowledge, and vulnerability-scanner false positives. | Case-study evaluation may limit generalizability; false positives and integration challenges remain concerns. | Improve vulnerability detection accuracy, reduce false positives, and strengthen automated integration of security tools into CI/CD pipelines. |
| X. Ramaj (2022)                           | DevSecOps-enabled risk management for critical infrastructures | Proposed a DevSecOps-based framework for managing cybersecurity risks throughout the development lifecycle of critical infrastructure systems.           | Highlighted the potential of DevSecOps for improving security and risk management in complex cyber-physical infrastructures.                   | Practical validation and large-scale empirical evaluation are limited.                                        | Develop and experimentally validate the framework in real-world critical infrastructure environments with continuous risk monitoring.          |

## V. CONCLUSION AND FUTURE WORK

A proactive method for combining security, development, and operations across the software development lifecycle is offered by DevSecOps. This survey examined Secure SDLC practices, shift-left security, continuous security testing, automated vulnerability management, Security as Code, CI/CD integration, compliance automation, and continuous monitoring. The reviewed technologies, including SAST, DAST, SCA, and Infrastructure as Code security scanning, demonstrate how automation can improve vulnerability detection, security consistency, deployment efficiency, and software resilience. Recent studies further indicate that DevSecOps can strengthen security in cloud-native, financial, distributed, and critical infrastructure environments while supporting faster and more reliable software delivery. The growing application of AI and ML offers additional opportunities for vulnerability identification, threat intelligence, security analysis, and adaptive security automation. However, organizational resistance, legacy systems, toolchain integration, scalability, performance limitations, false positives, and security knowledge gaps remain important



challenges. Overall, effective DevSecOps adoption requires collaboration, automation, continuous monitoring, and adaptable security practices.

Future research should focus on developing intelligent, scalable, and tool-independent DevSecOps frameworks for complex cloud-native and distributed environments. AI-driven vulnerability detection, automated compliance, adaptive security, and false-positive reduction require further validation through real-world deployments. Additional research should also investigate interoperability, performance optimization, continuous risk monitoring, and practical adoption across large organizations.

## REFERENCES

1. P. Martínez-Moreno, J. A. Vergara-Camacho, V. A. Luevano-Mondragon, and K. A. Jimenez-Martenez, "DevSecOps for Secure Software Development: A Systematic Literature Review of Practices, Benefits, and Adoption Barriers," *Int. J. Comb. Optim. Probl. Informatics*, vol. 17, no. 4, pp. 25–40, Aug. 2026, doi: 10.61467/2007.1558.2026.v17i4.1299.
2. J. Mason, Lopez, B. Dallas, and O. Kayali, "Secure Software Development Practices in Modern Applications," 2025.
3. C. S. Pareek, "Driving Agile Excellence in Insurance Development through Shift-Left Testing," *Int. J. Multidiscip. Res.*, vol. 3, no. 6, pp. 1–18, November-December, Nov. 2021, doi: 10.36948/ijfmr.2021.v03i06.22393.
4. X. Zhao, T. Clear, and R. Lal, "Identifying the primary dimensions of DevSecOps: A multi-vocal literature review," *J. Syst. Softw.*, vol. 214, p. 112063, Aug. 2024, doi: 10.1016/j.jss.2024.112063.
5. S. B. S. Ali, S. Tarakampet, and S. Tatavarthi, "Reusable, Secure, and Compliance-First CI/CD Pipeline Architectures for Regulated Enterprise Environments," in *2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISSET)*, Cairo, Egypt: IEEE, 2026, pp. 1–6, April. doi: 10.1109/ICAISSET66439.2026.11541732.
6. F. M. A. Hassan, S. R. Das, and M. Hussain, "Importance of Secure Software Development for the Software Development at Different SDLC Phases," Aug. 15, 2023. doi: 10.31124/advance.23947392.
7. K. K. Mohammed, "The Future is Cloud : Modernizing Big Data for the Cloud Era," *Int. J. Sci. Res. Eng. Trends*, vol. 11, no. 5, pp. 1–5, 2025.
8. M. Fu, J. Pasuksmit, and C. Tantithamthavorn, "AI for DevSecOps: A Landscape and Future Opportunities," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 4, pp. 1–61, May 2025, doi: 10.1145/3712190.
9. S. Arveti, M. Kalal, P. Bose, L. A. Yeruva, S. Inamdar, and R. K. Kanneganti, "Model Driven and Modular Software Architecture for Scalable Development," in *2026 International Conference on Computing Theory and Wireless Communications (ICCTWC)*, Ichalkaranji, India: IEEE, 2026, pp. 1–10, April. doi: 10.1109/ICCTWC68241.2026.11583718.
10. K. Jangiti, "Design and Validation of a Machine Identity Governance Framework for AI Agents in Multi-Cloud Environments," in *SoutheastCon 2026*, Huntsville, AL, USA: IEEE, 2026, pp. 1–6, February-March. doi: 10.1109/SoutheastCon63549.2026.11476363.
11. R. Ranka, S. Paranjpe, and R. Pachlor, "DevSecOps : Improving Software Development Lifecycle," vol. 4, no. 2584, pp. 175–180, 2024.
12. P. Abos, "DevSecOps for secure software development in the cloud," 2024.
13. A. Kudriavtseva and O. Gadyatskaya, "Secure Software Development Methodologies : A Multivocal Literature Review," 2022. doi: 10.48550/arXiv.2211.16987.
14. B. Singh, "Challenges and Solutions for Adopting DevSecOps in Large Organizations," vol. 7, no. 2, pp. 3673–3680, 2019.
15. J. Yankel and H. Yasar, "5 Challenges to Implementing DevSecOps and How to Overcome Them," Jun. 2023. doi: 10.58012/fywc-yq50.



16. J. Ounza, D. Odera, and M. Otieno, "Theory and practice in secure software development lifecycle: A comprehensive survey," *World J. Adv. Res. Rev.*, vol. 18, no. 3, pp. 053–078, Jun. 2023, doi: 10.30574/wjarr.2023.18.3.0944.
17. D. Venkatesh and M. Rakhra, "Agile adoption issues in large-scale organizations: A review," *Mater. Today Proc.*, Dec. 2020, doi: 10.1016/j.matpr.2020.11.308.
18. D. Parsons, R. Thorn, M. Inkila, and K. MacCallum, "Using Trello to Support Agile and Lean Learning with Scrum and Kanban in Teacher Professional Development," in *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, IEEE, Dec. 2018, pp. 720–724. doi: 10.1109/TALE.2018.8615399.
19. [H. J. Christanto and Y. A. Singgalen, "Analysis and Design of Student Guidance Information System through Software Development Life Cycle ( SDLC ) and Waterfall Model," vol. 5, no. 1, pp. 259–270, 2023, doi: 10.51519/journalisi.v5i1.443.
20. A. Tabassum, I. Manzoor, D. Shahid, A. Rida, and D. Imtiaz, "Optimized Quality Model for Agile Development: Extreme Programming (XP) as a Case Scenario," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 4, pp. 392–400, 2017, doi: 10.14569/IJACSA.2017.080453.
21. R. Amaro, R. Pereira, and M. M. da Silva, "Capabilities and metrics in DevOps: A design science study," *Inf. Manag.*, vol. 60, no. 5, p. 103809, Jul. 2023, doi: 10.1016/j.im.2023.103809.
22. H. Hoseinizade, A. Moini, and M. A. M. A. Kermani, "Designing synchronizer module in CMMS software based on lean smart maintenance and process mining," *J. Qual. Maint. Eng.*, vol. 29, no. 2, pp. 509–529, Apr. 2023, doi: 10.1108/JQME-10-2021-0077.
23. P. Sungu, N. Vincent, O. Nyangaresi, and S. Odhiambo, "Efficient Authentication Algorithm for Secure Remote Access in Wireless Sensor Networks," *J. Comput. Sci. Res.*, vol. 03, no. 04, pp. 43–50, 2021.
24. O. O. Abiona, O. J. Oladapo, O. T. Modupe, and O. Oyeniran, "The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline," *World J. Adv. Eng. Technol. Sci.*, vol. 11, no. 2, pp. 127–133, Mar. 2024, doi: 10.30574/wjaets.2024.11.2.0093.
25. S. Chittala, "Securing DevOps Pipelines: Automating Security in DevSecOps Frameworks," *J. Recent TRENDS Comput. Sci. Eng.*, vol. 12, no. 5, pp. 31–44, Nov. 2024, doi: 10.70589/JRTCSE.2024.5.5.
26. B. Singh, "Automating Security Testing in CI / CD Pipelines using DevSecOps Tools a Comprehensive Study," vol. IX, no. XII, pp. 631–652.
27. R. Korrapati, "Automating Compliance in CI/CD Pipelines: A Modern Software Development Framework," 2025. doi: 10.2139/ssrn.5139607.
28. A. Al Noman, S. O. Idowu, R. D. M. Kakanou, P. Ciancarini, and M. Ren, "An AI-Based Security Analysis Solution for DevSecOps," 2026, pp. 274–290. doi: 10.1007/978-3-032-09694-4\_22.
29. T. Abdiukov, "Automated security testing in DevSecOps pipelines: Integrating AI-based vulnerability discovery and compliance validation," *World J. Adv. Res. Rev.*, vol. 22, no. 1, pp. 2083–2093, Apr. 2024, doi: 10.30574/wjarr.2024.22.1.1083.
30. D. Velaga, S. C. Chava, K. Ponaka, R. K. Kanneganti, A. Gupta, and N. Mukhi, "DevSecOps-Driven Software Engineering for Secure and Rapid Deployment," in *2026 International Conference on Computing Theory and Wireless Communications (ICCTWC)*, IEEE, Apr. 2026, pp. 1–7. doi: 10.1109/ICCTWC68241.2026.11583617.
31. S. N. V, R. S. B, and J. Sathiamoorthy, "Integrating Security into CI/CD Pipelines: A DevOps Approach to Application Deployment," in *2026 Second International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*, IEEE, Jan. 2026, pp. 1–6. doi: 10.1109/ICISCoIS62701.2026.11447658.
32. S. Dasanayake, J. Senanayake, and W. Wijayanayake, "DevSecOps implementation for continuous security in financial trading software application development," in *2025 5th International Conference on Advanced*



- Research in Computing (ICARC), Belihuloya, Sri Lanka: IEEE, 2025, pp. 1–6, February. doi: 10.1109/ICARC64760.2025.10963292.
33. M. Chen, B. Liang, and X. Lu, “The Practice and Application of a Novel DevSecOps Platform on Security,” in 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), IEEE, Mar. 2024, pp. 558–562. doi: 10.1109/AINIT61980.2024.10581700.
34. M. A. Aljohani and S. S. Alqahtani, “A Unified Framework for Automating Software Security Analysis in DevSecOps,” in 2023 International Conference on Smart Computing and Application (ICSCA), IEEE, Feb. 2023, pp. 1–6. doi: 10.1109/ICSCA57840.2023.10087568.
35. X. Ramaj, “A DevSecOps-enabled framework for risk management of critical infrastructures,” in Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings, New York, NY, USA: ACM, May 2022, pp. 242–244. doi: 10.1145/3510454.3517053.

