www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(4), 2026, 25-40. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2026.v17i4.1299

DevSecOps for Secure Software Development: A Systematic Literature Review of Practices, Benefits, and Adoption Barriers

Patricia Martínez-Moreno[1], José Antonio Vergara-Camacho[1], Víctor Adrián Luévano-Mondragon[1] and Karla Alejandra Jiménez-Martínez[2]

[1] Facultad de Contaduría y Administración, Universidad Veracruzana, campus Coatzacoalcos, México
[2] Tecnológico Nacional de México/ ITS de Coatzacoalcos, México
Email address(es): pmartinez@uv.mx, jvergara@uv.mx, victorluevmon@gmail.com, kjimenezm@itesco.edu.mx

Abstract. Integrating security protocols early in the software development lifecycle (SDLC) has become increasingly critical, positioning DevSecOps as a vital industry standard. This proactive approach not only ensures regulatory compliance but also minimizes application vulnerabilities and accelerates the identification of potential threats. Through a Systematic Literature Review analyzing forty-two selected empirical studies, this study analyzes the contemporary landscape of DevSecOps implementation. The primary goal is to map out current practices to evaluate existing strengths, identify areas of opportunity, and guide future research avenues. Ultimately, the findings underscore that while DevSecOps builds robust cybersecurity defenses and fosters a collaborative, open organizational culture, its integration is frequently hindered by notable barriers. Specifically, companies struggle with workforce resistance, broader cultural shifts, and the technical intricacies of deploying novel security mechanisms.

Keywords: DevSecOps; secure software development; software development life cycle; systematic literature review; security by design; continuous security; software security practices; cybersecurity resilience; threat detection; DevSecOps adoption barriers; organisational culture; security automation.

Article information
Received: Jan 17, 2026
Accepted: May 19, 2026

1 Introduction

Building software represents an inherently intricate and heavily systematized undertaking, consisting of numerous distinct phases that teams must execute with absolute precision to reach their targeted outcomes. According to Pressman and Maxim (2019), engineering procedures should never be interpreted as inflexible mandates for building digital products. Rather, they function as highly versatile frameworks, granting developers the freedom to identify and utilize the exact combination of activities and duties best suited for their specific assignment. Consequently, throughout recent decades, these developmental practices have experienced substantial transformations, dramatically altering both the underlying technologies used and the ways engineering groups structure and supervise their daily workloads.

Historically, initial software engineering efforts depended heavily on highly linear frameworks, including the spiral, V, and waterfall models. While these structures served a purpose during their era, their inherent rigidity made accommodating shifting project requirements exceptionally difficult, a limitation that routinely caused inflated budgets and extended delivery timelines. To overcome these hurdles, industry witnessed the rise of Agile paradigms which completely transformed development practices. These modern tactics consistently prioritize iterative cycles for defining, building, and deploying digital products. As Ian S. (2011) highlights, such strategies excel in environments characterized by rapidly fluctuating prerequisites during the design phase. Consequently, engineering cohorts can now produce superior technological solutions at an accelerated pace.

Building upon these iterative successes, the tech sector subsequently embraced DevOps, an integration of novel tooling, operational habits, and cultural shifts. Deeply rooted in Agile philosophies, this paradigm champions expedited feedback loops and compressed release schedules, effectively forging a sustainable rhythm of ongoing refinement (Borja V., 2021).

The paradigm shift, nevertheless, continues to progress. An escalating contemporary necessity to embed defensive measures throughout the entire pipeline has driven the transition toward the DevSecOps framework. Driven by the imperative to meld operational strategies with robust protection, this modern methodology diverges from basic DevOps by anchoring cybersecurity as a continuous, foundational element starting right from the initial planning phases. Consequently, this progression prompts vital inquiries: What concrete advantages does the DevSecOps model offer throughout a project's lifecycle, and what specific obstacles might organizations encounter during its adoption?

2 The DevSecOps field

To address the escalating demand for uninterrupted security integration throughout the software creation pipeline, the DevSecOps methodology was introduced. Rooted in Agile principles, this paradigm relies on collaborative teams operating in brief, iterative sprints to deliver progressive system enhancements.

Numerous scholars have explored this domain, yielding crucial foundational insights. For instance, an investigation by Morales et al (2020) evaluated the consequences of inadequate DevSecOps execution by gathering practical takeaways from a corporate project blending Agile and DevSecOps frameworks. The primary findings demonstrated that executing DevSecOps incompletely spawned vulnerabilities, ultimately causing schedule overruns and complicating overall system administration. Conversely, Scanlon & Morales (2022) examined the educational takeaways derived from a company's shift toward DevSecOps and Agile structures. This analysis pinpointed critical roadblocks, notably the buildup of technical debt and an absence of unified defensive testing, while offering strategic advice for smoothing subsequent organizational shifts. Furthermore, research by Cankar et al. (2023) sought to bolster programming safety through the deployment of specialized utilities. The core conclusions revealed that instruments like LOMOS and IaC Scan Runner significantly boost system reliability across both the planning and deployment stages.

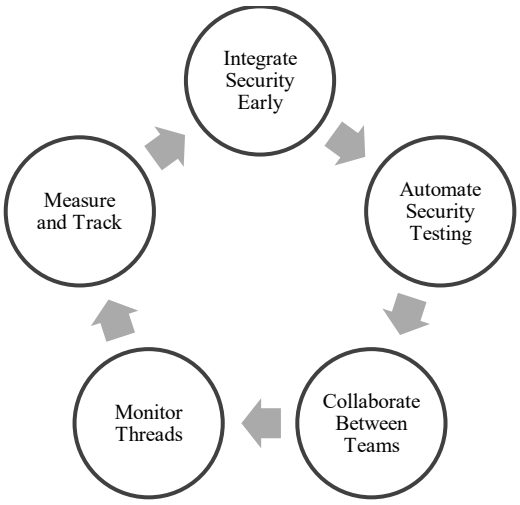

Fig. 1. DevSecOps lifecycle.

Another investigation by Morales et al. (2020) concentrated on evaluating the contemporary DevOps environment to harmonize theoretical studies with real-world corporate applications. This endeavor utilized a multivocal literature strategy, synthesizing

formal scholarly articles with gray literature. The resulting observations highlighted vital elements while emphasizing how crucial it is to comprehend the viewpoints of active industry specialists.

Lastly, an inquiry conducted by Fabiola (2024) sought to offer actionable remedies for expansive secure software construction by hosting focus groups and collaborative workshops with specialists across various disciplines. The primary outcomes categorized five distinct hurdles related to continuous security programming and mapped out four vital trajectories for upcoming academic exploration. Collectively, this foundational literature illustrates that while embracing DevSecOps holds immense potential, organizations confront substantial obstacles regarding its practical execution and overall efficacy across varied operational settings. The standard DevSecOps lifecycle (Fig. 1) serves as a foundational blueprint for embedding defensive measures into programming from the very beginning, acting as the central pillar for security enhancements within ongoing engineering pipelines. Consequently, identifying the specific circumstances and variables that dictate the triumphant integration of DevSecOps holds tremendous academic and practical value. Unquestionably, this methodology has firmly cemented itself within modern software engineering practices.

3 Research questions

The primary aim of this systematic literature analysis is to present a comprehensive snapshot of the contemporary empirical research surrounding DevSecOps implementation. By doing so, this document assists in mapping out current advantages and potential gaps, thereby laying the groundwork for subsequent academic investigations. Furthermore, this evaluation delivers a deeper comprehension of how DevSecOps influences and empowers the creation of well-protected software architectures. This review aims to resolve the specific analytical inquiries detailed below (Table I):

Table 1: Research questions and their motivation

ID	Research question	Motivation
1	What is the importance of integrating security early into the DevOps/DevSecOps pipeline?	Explore the most critical security integration aspects by implementing DevSecOps into the SDLC.
2	How does DevSecOps integrate security into the software development lifecycle?	Identify the key elements of DevSecOps for achieving security integration into the SDLC.
3	What are the main barriers companies face when adopting DevSecOps?	Identify the main barriers or difficulties that companies have faced in adopting DevSecOps within their software development processes.
4	What secure development practices are implemented based on DevSecOps?	Identify the strategies and/or practices that have facilitated successful DevSecOps implementation.
5	What tools are used in DevSecOps to integrate security early in the software development lifecycle?	Identify the most helpful software solutions or tools for integrating security from the early stages of the SDLC.
6	What are the main challenges when implementing DevSecOps in microservices and container environments?	Identify the complexities of security in containers and distributed environments, where traditional security techniques may be insufficient or obsolete.
7	Why is the “Shift Left” approach insufficient on its own, and how does ‘Shift Right’ complete the lifecycle?	Explore why focusing solely on the early stages (Shift Left) is insufficient against active cloud workload attacks and how Shift Right ensures the desired security state through continuous monitoring and response.

4 Review methodology

The methodology used was based on the guidelines proposed by Kitchenham and Charters (2007) for conducting systematic reviews in Software Engineering. The review was conducted in different stages: development of the review protocol, identification

of inclusion and exclusion criteria, search for relevant studies, critical appraisal, and data extraction and synthesis. In the remainder of this section, we describe the details of these stages and the methods used. In December 2025, a literature search was conducted in specialized databases, focusing on a five-year retrospective period; Table 2 describes the process of selecting articles for the systematic review, from initial identification to final inclusion; in December 2025, a literature search was conducted in specialized databases, focusing on a five-year retrospective period.

Studies were eligible for inclusion in the review if they presented empirical data on building secure software using the DevSecOps approach and met minimum quality requirements. Inclusion of studies was not limited to any specific type of intervention or outcome measure. The systematic review included qualitative and quantitative research studies published between 2019 and 2025. Only studies written in English were included.

Studies whose primary focus was not on secure software development through DevSecOps or did not present empirical data were excluded. Articles without a research question or research design and articles based solely on expert opinion were also excluded. The inclusion and exclusion criteria applied in the search are described in Fig. 3.

4.1 Search strategy

The search strategy included electronic databases. Various versions of the search strings were tested, primarily in the IEEE Xplore, ACM Digital Library, and Wiley Online Library databases. The following search string (Fig. 2) was considered to retrieve an adequate number of articles addressing the DevSecOps approach or DevOps security, analyzing how security is integrated from the early stages, covering practices and tools, and documenting difficulties and challenges. All searches were conducted using the article title, abstract, and keywords for the study.

```
("DevSecOps" OR "DevOps" OR "Secure Software Development")
AND
("security integration" OR "shift left security" OR "early security" OR "security by
design" OR "built-in security" OR "security pipeline" OR "shift right" OR "runtime
security" OR "continuous monitoring" OR "observability")
AND
("practices" OR "methods" OR "techniques" OR "tools" Or "automation" OR "CI/CD" OR
"continuous integration" OR "continuous deployment" OR "microservices" OR
"containers" OR "kubernetes" OR "orchestration")
```

Fig. 2. Search string.

Once the inclusion criteria (IC) and exclusion criteria (EC) were defined, as described in Fig. 3, they were distributed into four application stages. These stages were used to filter the sources obtained in each academic database to ensure they met the required approach. These stages were divided into the following selection stages:

- Stage 1: Identify relevant studies from 2019–2025 in English
- Stage 2: Exclude non-empirical, duplicate, and non-full-access studies
- Stage 3: Exclude articles based on the abstract
- Stage 4: Obtain articles that answer at least one question

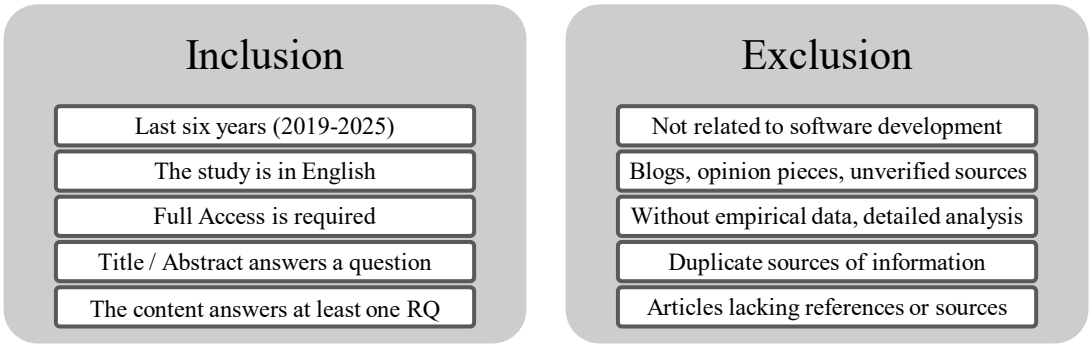

Fig. 3: Inclusion and exclusion criteria

Fig. 4 illustrates the number of articles obtained at each stage of the search and selection process. The steps followed during this inclusion and exclusion phase were as follows: Stage 1: Identify relevant studies from 2019 to 2025 in English; Stage 2: Exclude non-empirical, duplicate, and non-full access studies; Stage 3: Exclude articles based on the abstract; Stage 4: Obtain articles that answer at least one question. At each step, articles that clearly did not fall within the scope of DevSecOps were excluded. During the selection stages, the number of primary studies was reduced from 467 to 42, and they were subjected to evaluation and analysis, the results of which are presented in the following section.

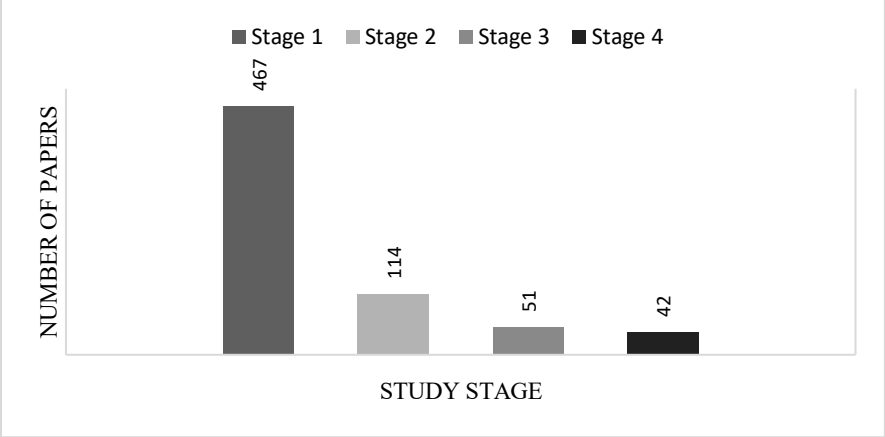

Fig. 4: Number of studies selected by stage

4.2 Quality assessment

The studies retrieved in stage 3 were individually assessed using the quality control instrument proposed by Dyba˚ and Dingsøyr (2008), which guides an approach with questions that assess internal validity, external validity, clarity, repeatability, and bias. The instrument was used to establish the following questions. Each question was scored on a scale: 1 = Yes, 0.5 = Partially, No = 0:
Q1 Is the objective of the study clearly defined?
Q2 Is the context adequately described?
Q3 Is the methodology adequate and clearly explained?
Q4 Are threats to validity discussed?
These criteria allowed us to obtain a measure of the degree of confidence that the findings of a particular study could significantly contribute to this review on DevSecOps. Of the 51 articles assessed for quality, 9 articles were excluded, leaving 42 primary studies for data extraction and synthesis.

Table 2: Methodological workflow: Article selection process

Stage	Actions performed	Result (N)
1	Database search (IEEE Xplore, ACM DL, Wiley OL) using specific search strings for the 2019-2025 period.	467
2	Removal of duplicates, non-empirical studies, and articles without full-text access.	114
3	Review of titles and abstracts to ensure alignment with the research scope.	51
4	Quality assessment and evaluation of the ability to answer Research Questions (RQs). (9 articles were excluded during this stage).	42

5 Results

Forty-two quality articles were retrieved for the study topic. This section first describes the general results of the literature review and then answers the research questions by analyzing the collected data on DevSecOps (Table 3).

Fig. 5 shows the years the analyzed studies were published, highlighting the source of the relevant retrieved studies in a colored area. It can be seen that the majority of articles were published in more recent years, demonstrating the importance of the

DevSecOps approach within the software development process. It is worth mentioning that some analyzed articles only mentioned the term DevOps but addressed security as a fundamental aspect to be implemented from the early stages of software development.

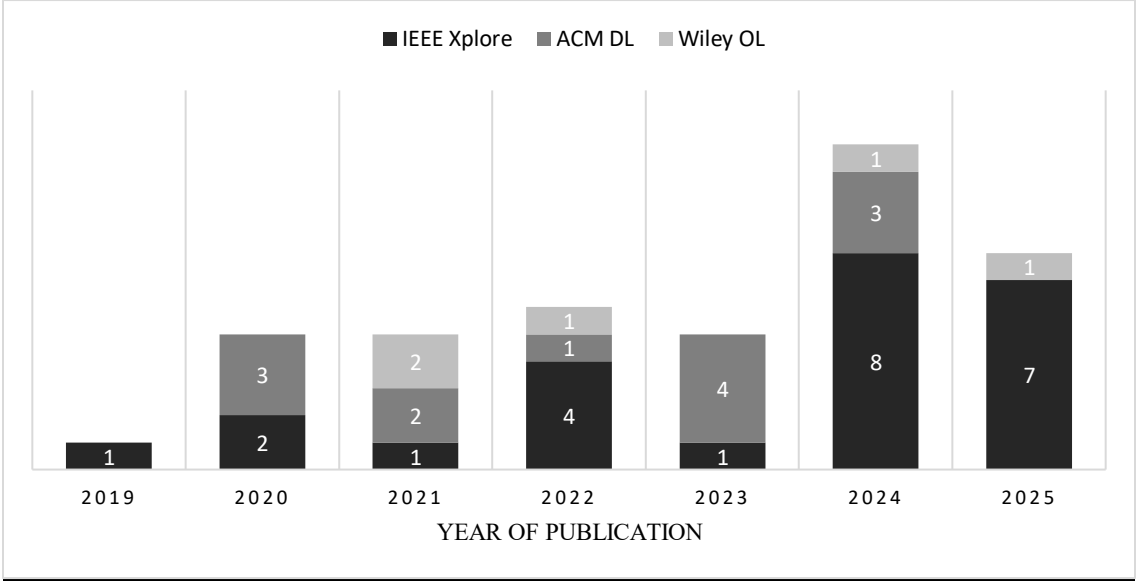

Fig. 5: Number of studies per year

RQ1: What is the importance of integrating security early into the DevOps/DevSecOps pipeline?

Based on the evidence collected, we investigated the importance of integrating security early into software development, considering DevSecOps/DevOps approaches. As a result, we identified that 16 of 42 articles described the importance of integrating security (a quality attribute) early, and the majority recognized the DevSecOps approach as an essential strategy that favors the integration of this attribute into the software development lifecycle (SDLC). Adopting DevSecOps is fundamental to transitioning from traditional models (such as waterfall) to an agile environment based on Kubernetes (Soonhong et al, 2025).

Security is an essential attribute of quality software Akbar et al. (2024); the DevSecOps approach refers to the integration of security practices into the DevOps process from the earliest stages of a project; this approach is an extension of Lean-Agile principles, and its implementation depends on its success (Richard T. 2021; Maysa et al, 2025). The reviewed studies highlight the integration of security from early phases of the life cycle through DevSecOps practices (Cheenepalli et al., 2025; Hermann et al., 2025; Nagasundari et al., 2025; Emmanouil et al., 2024; Agung and Herman, 2022; Michaelet al., 2022; Akbar et al., 2024) to establish a security-centric approach, improve team collaboration and leverage technologies, as mentioned by Yulianto (2024), lead to more secure and resilient software systems, emphasizing the importance of seamlessly integrating security practices into the development process.

In Nagasundari et al. (2025), it is mentioned that "securing the software delivery pipeline is as important as securing the software being delivered," which has led to integrating security from the pipeline, where product development begins. Furthermore, an important step toward improving cybersecurity defense is acquiring a deep understanding of DevSecOps (Nagasundari et al., 2025), since as cyberthreats become more sophisticated and frequent, the importance of this approach in the SDLC is undeniable (Emmanouil et al., 2024; Shripad et al., 2021). Integrate security from outset enables the creation of automated pipelines where source code and construction artefacts are systematically scanned, ensuring protection against known attack vectors without the need of constant human interfaces (Jamal M. and Joel C., 2021).

The literature agrees that DevSecOps has helped developers shorten the overall software development lifecycle and, as a result, decreased time to market (Hasan Y. and Sam E., 2022). However, like any emerging technology, the field offers many challenges that have not been faced with traditional software systems.

Mengxia et al. (2024) report two important aspects of DevSecOps: on the one hand, it favors a balance between efficiency and security in projects, and on the other hand, it can create a secure and open culture, improving communication and collaboration between development, operations, and security teams. Specifically, in the study (Mengxia et al., (2024), they found that

DevSecOps represented a significant breakthrough in accelerating release iterations and reducing security vulnerabilities in production environments, concluding that companies can complete the integration of security prevention and control measures and security detection tools at all stages of requirements planning. These emerging technologies have demanded the need to put security at the center of the code-to-container workflow, giving rise to DevSecOps paradigms (Shripad and Laura L., 2021).

In Cankar et al. (2023) it is highlighted that Runtime Application Self-Protection (RASP) allows for early detection of attacks and rapid mitigation of them; this is crucial because traditional measures, such as web application firewalls, have difficulty accurately identifying vulnerabilities without constant manual adjustments.. In modern architectures such as microservices, considering security as an afterthought (after development) is inefficient and even prohibitive due to the distributed nature and expanded attack surface of these applications (Priyanka et al., 2022).

In summary, DevSecOps has the primary purpose of helping development teams deliver more secure software, through a repeatable and adaptable process that ensures security is applied from the early stages of the software development lifecycle (Akbar et al., 2024); only when everyone participates in DevSecOps can security be guaranteed (Xin et al., 2023); this approach aims to solve problems as soon as they arise in the software, before they become complex to fix (Maysa et al., 2025).

RQ2: How does DevSecOps integrate security into the software development lifecycle?

The differences between DevSecOps and traditional practices primarily lie in the introduction of frameworks that facilitate a better culture of collaboration (Muntaha et al., 2022); with DevSecOps, security is the responsibility of all IT team members (including development, testing, operations, maintenance, and security teams (Wang et al., 2022).

For the successful integration of security practices into the SDLC, in Cheenepalli et al. (2025), it is emphasized that the commitment of the company's leadership is essential, and, in the specific case of its integration into microservices, (Yulianto and Gerard, 2024) tells us that it makes it easier to address security challenges and take advantage of the security services offered in the cloud.

In Feio et al. (2024), it is emphasized that the main component of DevSecOps is the CI/CD pipeline, composed of a set of consecutive phases that define various activities and tools to facilitate the automation of software development. The same study identified the common stages in most pipelines: plan, program/develop, build, test, release, deploy, operate, and monitor (Feio et al., 2024). Hasan Y. (2022) also concludes that configuration management is essential for any DevSecOps pipeline, especially where test and production environments can be highly complex. The authors in Solaimalai (2025) point out that DevSecOps uses machine learning models to enable real-time threat detection and risk assesment, reducing the manual intervention required in the SDLC.

In summary, integrating security through DevSecOps requires mature, disciplined, and continuously improving development of team knowledge, skills, and motivation (Akbar et al., 2024), thus ensuring that security considerations are integrated from the beginning of the project (Natalie et al., 2024; Díaz et al., 2019).

RQ3: What are the main barriers companies face when adopting DevSecOps?

In Cheenepalli et al. (2025), technical complexity was found to be one of the main barriers to the implementation of DevSecOps, followed by other factors such as resource limitations (Hermann et al., 2025) and lack of experience. According to the authors in Haverinen et al. (2024), adopting DevSecOps practices as part of the software development process has proven to be tedious and costly in practice. There are several barriers to implementing security with DevSecOps. However, the human factor is one of the main factors that influences the effectiveness or failure of a security system, as highlighted in (Akbar et al., 2024), because those involved in the DevSecOps process can make mistakes due to a lack of security skills and knowledge.

There are undoubtedly challenges to adopting DevSecOps. In Mengxia et al. (2024), various barriers are mentioned, such as culture, processes, and technology, highlighting the need to train security experts with the ability to incorporate security methods into the company's actual workflow properly. The authors in Natalie et al. (2024) conclude that there are several challenges in implementing this approach: cultural change, tool integration, and a skills gap. In Lazarus et al. (2024) they argue that while DevSecOps as a concept seems simple, it lacks precise and structured guidance on how programs should implement this process. The technological gap between security tools and software delivery technologies is one of the main barriers to integrating security

into a pipeline. Given this situation, Roshan et al. (2021) identified that open source products are emerging to meet DevOps needs by releasing new generations of security tools such as IAST and RASP.

It is challenging to fully incorporate security controls into DevSecOps using outdated techniques. In Morales y Yasar (2023), it is concluded that a culture of traditional mindsets hinders changes in the current development of pipelines. However, this can be partially addressed through ongoing training and awareness of the latest trends and their benefits. One of the barriers identified in Cankar et al. (2023) is the propensity for human error in IaC scripts, such as the exposure of credentials or the application of incorrect or outdated configurations. In Debi and Gail (2020), it is mentioned that software development initiatives such as DevSecOps are sometimes seen as synonymous with rejecting the "extra development efforts" required by security and are considered overhead. Security culture is a key aspect (Maysa et al., 2025; Xin et al., 2023) within the organizational structure, and it should be included as a natural aspect of the developer's routine rather than seen as a separate add-on (Debi and Gail, 2020), which implies involving the security team throughout the development process (Akbar et al., 2025).

As described in the first question, the importance of DevSecOps adoption is undeniable. But in Shripad and Laura (2021) argues that it is limited to the CI/CD or shift left domain, which could be insufficient to maintain the desired security state for applications in the face of new vulnerabilities, malicious attacks on the cloud workload, or changes in regulatory policies, which leads to transitioning security in DevSecOps through shiftleft and shiftright. In addition, it is complex to fully incorporate security controls in DevSecOps using obsolete techniques (Maysa et al., 2025). The authors in Akbar et al. (2024) mention that DevSecOps barriers are associated with tools, methods, and organizational work culture, the findings in Mao et al. (2020) identify that the primary external challenges are the lack of DevSecOps experts, tools and having consolidated DevSecOps solutions, and internally, cultural resistance, high costs and organizational structure (for example, the lack of a security leader). In Nicholas et al. (2022) the authors conclude that existing DevSecOps solutions still require considerable manual intervention to function, which limits their ability to cover live threats.

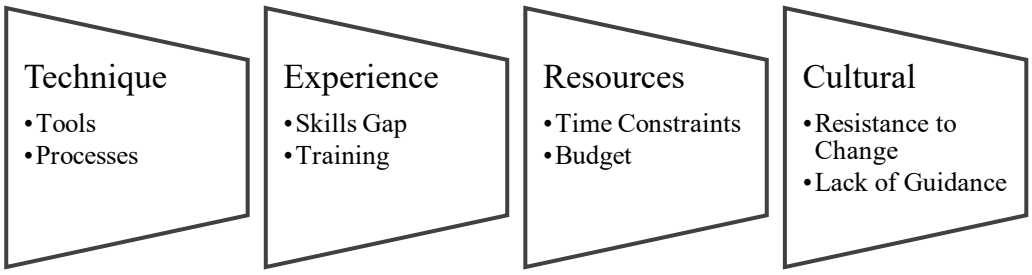

Fig. 6: Barriers to adopting DevSecOps

RQ4: What secure development practices are implemented based on DevSecOps?

Among the findings in Cheenepalli et al. (2025), it is mentioned that emerging technologies such as AI and ML favor secure development practices. This aspect is also highlighted by Yulianto and Gerard (2024) when arguing that "the incorporation of AI into DevSecOps practices represents a proactive and innovative strategy to strengthen the security of software development." This aspect significantly benefits the expansion of security measures in the SDLC (Yulianto and Gerard, 2024). In Muntaha and Imad (2022), it is mentioned that best security practices should be integrated into IaC processes by design, enabling organizations to achieve security success by combining threat model analysis with a DevSecOps workflow (Nagasundari et al., 2025).

On the other hand, implementing the integration of Snyk and StackHawk in the CI/CD pipeline enables early vulnerability detection (Manohar et al., 2023); the study Natalie et al. (2024) identifies these key practices for secure development: CI/CD, vulnerability automation and scanning, secure coding, container security, and decentralized security. There are emerging technologies that support secure development best practices. In Natalie et al. (2024), it is mentioned that cybersecurity AI allows for the automation of threat analysis, anomaly detection, and response to potential cyberattacks. In Yu et al. (2024), a framework based on DevSecOps and AIOps was presented to improve security, combining continuous monitoring, machine learning-based threat detection, and intelligent response orchestration, significantly improving security automation's accuracy and speed.

The research Roshan et al. (2021) concludes that in order to successfully integrate security into DevOps (DevSecOps), it is not enough to use adequate security tools; professionals must adopt appropriate workflows that integrate security tools seamlessly. Pipelines are a fundamental component of a DevSecOps strategy and are widely used across all industries (Morales y Yasar,

2023). They can also include configuration management and test environment generation (Richard, 2021). In DevSecOps, full-system testing focuses on functional, regression, and security testing performed continuously (Michael et al., 2022).

In summary, there are various secure development practices through DevSecOps, but there are common aspects to consider in research and the professional field. The authors in Hasan and Sam (2022) highlight the following guidelines for implementing DevSecOps pipelines regardless of the type of system to be developed: culture, automation, Infrastructure as Code (IaC), monitoring, and the DevSecOps platform (communication, supervision, automation). As cited in Debi and Gail (2020), the key to successful DevSecOps practice is: culture, automation, sharing, and measurement, which Emerson (2020) complements, indicating that continuous feedback and testing are necessary for a high-performing DevSecOps team.

The authors in Xin et al. (2023) found that DevSecOps capabilities include shifting security to the left and continuous security, where shifting to the left, in addition to introducing security activities in the initial stage of development, involves integrating security throughout the DevOps life cycle, and with respect to continuous security, it focuses on learning and continuous improvement of projects. In Xin et al. (2023) it is recommended to apply the principle of least privilege and separation of duties, establishing a strict separation of roles between Development (Dev), Security (Sec) and Operations (Ops), where each party has permissions limited to its specific tasks.

RQ5: What tools are DevSecOps using to integrate security early in the software development lifecycle?

Among the tools reported, Jenkins (Feio et al., 2024), SonarQube, OWASP ZAP, StackHaw (Manohar and Salaja, 2023) for CI/CD pipelines, Snyk, GitLab CI/CD and Docker (Emmanouil et al., 2024; Natalie et al., 2024), and Anchore for container analysis stand out. DevSecOps is mainly based on changes in mentality that favor collaboration and communication between team members and others. It introduces tools for automation and integration (Muntaha and Imad, 2022). The same study tells us that continuous deployment tools refer to the automatic redistribution of a developer's changes, controlling the configuration and management of the deployment environment. The literature highlights the DevSecOps pipeline that integrates static and dynamic testing using GitLab for code and tools such as Njsscan and OWASP-ZAP for testing functions (Nagasundari et al., 2025).

AI-based security frameworks such as ATT&CK (or the Kubernetes ecosystem (Muntaha and Imad, 2022; Nagasundari et al., 2025, Emmanouil et al., 2024) can improve security by establishing environments to address cybersecurity challenges. Other suggested aspects include static code analysis, code analysis during continuous integration, and anomaly detection in the SDLC (Yulianto and Gerard, 2024). In addition to the tools described above, the authors in Wang et al. (2022) indicate that a set of tools is needed that significantly improves the efficiency of collaboration, knowledge sharing, and self-service capabilities. The study Manohar et al. (2023) mentions that selecting the right tools is crucial for the effective implementation of security analysis in a DevSecOps CI/CD pipeline (Richard, 2021), such as the combination of Snik (SAST) and Stack-Hawk (DAST) for security analysis in a DevSecOps pipeline (Maysa, 2025).

In Agung and Herman (2022), they conclude that DevSecOps has been successfully implemented using GitLab and Docker tools integrated with the deployment server. There are many tools to support early security integration; however, the authors in Roshan et al. (2021) conclude that there is an urgent need to devote more attention and resources to developing and evaluating security tools suitable for DevSecOps. In summary, there are various tools to support the integration of security through DevSecOps. In this sense, evidence shows the great importance of automation in software engineering practices (Maysa and Iqbal, 2025), such as continuous automation, continuous delivery, and continuous feedback, in integrating security controls in the early SDLC stages.

RQ6: What are the main challenges when implementing DevSecOps in microservices and container environments?

The implementation of DevSecOps within microservices and containerized environments introduces a unique set of challenges primarily driven by the architectural shift toward distributed systems. A fundamental concern is the increased attack surface inherent in microservices, where the multiplicity of distributed interfaces and endpoints exposes significantly more data than traditional monolithic structures (Priyanka et al., 2022). This complexity is further exacerbated by the "large attack surface" created by the evolutionary nature of services, making consistent security enforcement a moving target.

Furthermore, the effectiveness of security monitoring is often hindered by the limitations of traditional tooling and the ephemeral nature of cloud-native components. Solaimalai (2025) notes that conventional security tools frequently fail to adapt to the scale and dynamism of these environments, leading to an excess of false alarms and severe alert fatigue among staff. Consequently, because functions and containers often have a very short lifespan, maintaining visibility becomes difficult; as Barrak et al. (2025)

explain, the lack of unified logging and the rapid destruction of resources impede both real-time monitoring and post-attack forensic analysis.

In contrast to traditional perimeter security, managing access during execution presents critical technical hurdles regarding secrets and runtime protection. In environments where services are constantly created and destroyed, the static management of credentials becomes risky and insufficient, leading to problems involving network delays and service availability (Patel et al., 2025). Moreover, this is compounded by a "runtime protection gap," where Aditya and Muhamad (2025) identify a disconnect between static testing prior to deployment and the active protection required to defend against zero-day attacks during execution.

Ultimately, organizational and scalability issues remain a significant barrier to secure orchestration. Even when automated pipelines are utilized, systems remain vulnerable to internal threats, such as attackers possessing administrative privileges within a cloud service provider (Jamal and Joel, 2021). As systems grow, the sheer volume of components creates a scalability crisis; accordingly, Verderame et al. (2023) highlight that evaluating the security of thousands of individual containers is not a scalable practice without highly advanced, automated orchestration strategies.

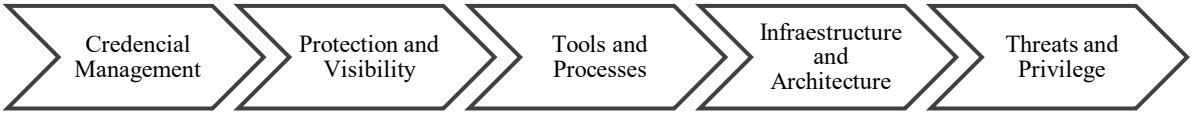

Fig. 7: Challenges of implementing DevSecOps.

RQ7: Why is the "Shift Left" approach insufficient on its own, and how does 'Shift Right' complete the lifecycle?

The "Shift Left" approach (shifting security to the beginning of development) is insufficient on its own because it leaves a critical security gap during execution, which is covered by the "Shift Right" approach (monitoring and protection at runtime). By focusing security on the early stages (planning, coding, and building), "Shift Left" has some limitations.

According to Aditya and Muhamad (2025), most current DevSecOps tools are static or pre-runtime, creating a critical protection gap against zero-day attacks or context-specific attacks that occur while the application is running. The authors in Barrak et al. (2025) mention that traditional static (SAST) and dynamic (DAST) testing are not directly integrated with the application and therefore cannot detect threats that arise in the runtime phase. In this context, Jamal and Joel (2021) mentions that continuous integration and deployment (CI/CD) processes generally do not address the security of runtime artifacts, especially their confidentiality when executed on third-party infrastructure, such as public cloud providers, and Patel et al. (2025) highlights that automated "Shift Left" pipelines do not protect against attackers with administrative privileges (internal threats), such as the staff of a cloud service provider.

The importance of "Shift Right" within the DevSecOps lifecycle lies in the implementation of continuous protection and monitoring once the application is in production. In this regard, Aditya and Muhamad (2025) mentions that Runtime Application Self-Protection (RASP) operates from within the application process, allowing it to observe actual behavior and take preventive measures against malicious inputs without relying on external layers; meanwhile, Barrak et al. (2025) and Morales et al. (2024) conclude that the cycle is completed by the monitoring phase.

Finally, to achieve proactive defense, the authors in Aditya and Muhamad (2025) mention that by integrating mechanisms such as RASP or trusted execution environments, real-time defense is achieved that reduces the window of exposure and improves detection accuracy without compromising development agility. With AI-based anomaly detection tools (such as LOMOS), it is possible to identify unknown problems and classify potential threats, allowing teams to focus on the highest-risk events (Nicholas et al., 2022) (Fig. 8).

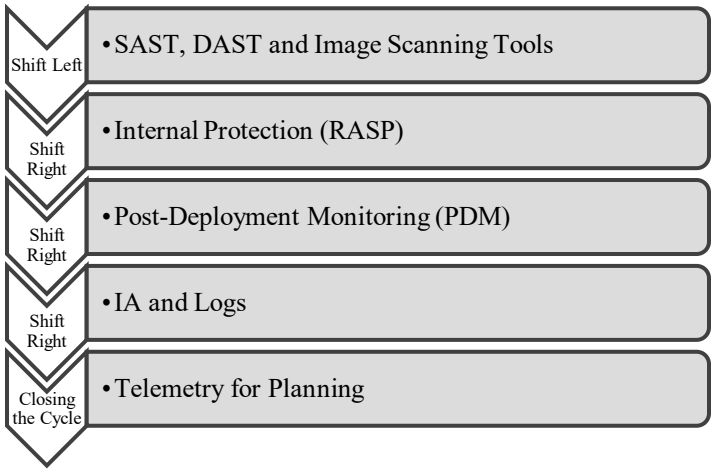

Fig. 8: DevSecOps Cycle: Shift Left and Shift Right

Table 3: Summary of the main findings

Category	Key Findings	Implies
Integrate security from early stages (RQ1)	53% consider it a key factor. Reduction of vulnerabilities. Accelerate delivery cycles. Enhance the culture of continuous safety.	Accelerate response capacity against threats. Lower costs for fault correction. Improves communication.
Integrate security in the SDLC (RQ2)	CI/CD as the central axis. Safety as a shared responsibility of the team. Use of IaC practices, testing. Requires organizational maturity and leadership.	Change the traditional scope from “security to the end” to "security at the beginning". Enables the detection and mitigation of risks at every stage of the process.
Barriers to DevSecOps adoption (RQ3)	Technical complexity and resource constraints. Security skills gap. Cultural resistance. Lack of seamless integration between tools. Absence of clear guidelines.	Requires investment in training. Needs redesign of processes and organizational culture. The lack of experts hinders effective adoption.
Secure development practices (RQ4)	Automated scanning. Security integration in IaC. Continuous monitoring. Use of Artificial Intelligence. Container and microservices security. Continuous testing.	Increases accuracy and speed in risk identification. Reduces manual dependencies. Facilitates consistent environments for development and testing.
DevSecOps Tools (RQ5)	Jenkins, GitLab CI/CD. Docker, SonarQube. OWASP ZAP, StackHawk. IaC tools. AI-based frameworks.	Facilitate the integration of static, dynamic, and automated testing. Provide traceability, quality, and security throughout the pipeline.
Identify the complexities of security in containers and distributed environments (RQ6)	Credential management, protection an visibility, tool and processes, infrastructure and architecture, threats and privileges.	Risk of leaks in code or repositories, traditional tools are ineffective against the dynamism of the cloud, fragmented logging between components, massive increase in the attack surface.
Shift Left is not enough to guarantee safety and how Shift	SAST, DAST and image scanning tolos, post-deployment monitoring, telemetry for planning, RASP, IA and logs.	Dependency scanning, threat modeling, anomaly monitoring, log analysis (AI).

Right completes the desired safety (RQ7)		

5.1 Cross-cutting findings

The cross-analysis of the 42 primary studies reveals that DevSecOps should be understood as a multi-layered socio-technical transformation rather than a mere extension of DevOps with security tools. Across all research questions (RQ1–RQ7), three overarching patterns emerge.

First, DevSecOps operates as an integrated lifecycle model in which shift-left (preventive) and shift-right (runtime) mechanisms are increasingly interconnected. Security is no longer confined to early development stages; instead, runtime monitoring, telemetry, and vulnerability intelligence feed back into CI/CD pipelines, forming a continuous security feedback loop.

Second, automation constitutes the structural backbone of DevSecOps. CI/CD security gates, vulnerability scanning, container security, policy-as-code, and AI-driven detection enable scalability and consistency in high-velocity environments. However, automation also introduces orchestration complexity, demanding careful governance and architectural alignment.

Third, organizational and cultural factors remain decisive. Despite technological maturity, barriers such as skill gaps, siloed teams, and resistance to shared responsibility persist. Evidence consistently indicates that successful DevSecOps adoption requires synchronized evolution across governance, tooling, architectural design (particularly in cloud-native and microservices contexts), and human capabilities.

Additionally, AI is emerging as an augmentation layer—enhancing anomaly detection and predictive analysis—but remains transitional, with limited large-scale empirical validation. Overall, the findings support conceptualizing DevSecOps as a continuous, automation-driven security orchestration ecosystem embedded within organizational culture and distributed architectures.

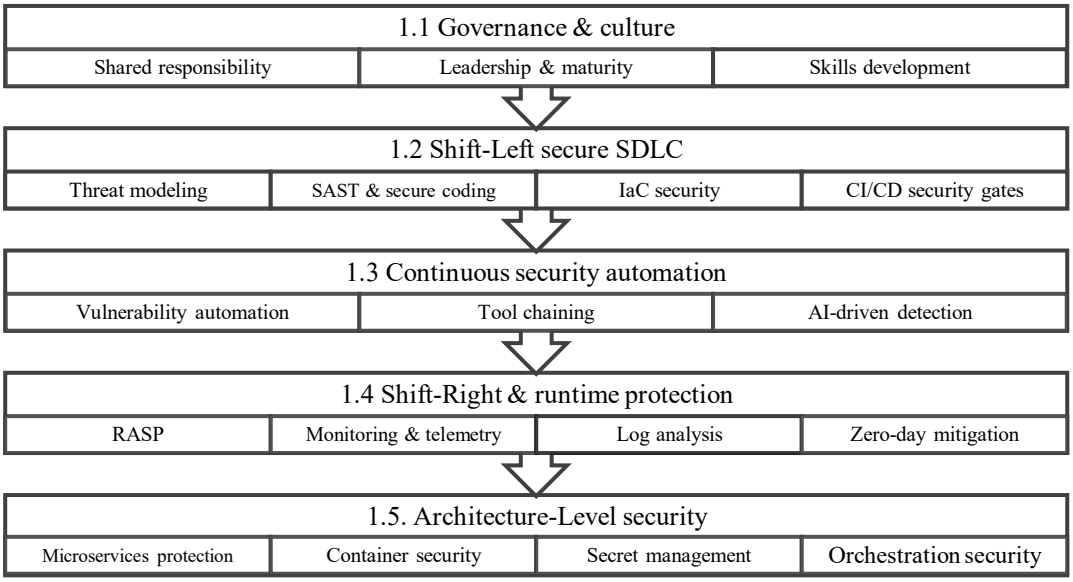

Fig. 9: Conceptual taxonomy of DevSecOps practices

6 Validity of the study

To ensure the integrity of the findings and mitigate biases during the data identification and extraction phases, various mitigation strategies were implemented. Initially, an exploratory terminological phase focused on the DevSecOps ecosystem was conducted. The objective was to establish a precise technical vocabulary to refine the search string. Following the guidelines proposed by Zhang et al. (2011) regarding manual search methodologies, the automated search string was validated against a predefined set of

known "seed" studies across databases such as IEEE Xplore, ACM Digital Library, and the Wiley Online Library. This calibration process ensured that the final query was sufficiently robust to retrieve relevant literature while minimizing excessive noise.

Despite the methodological rigor applied, inherent threats to validity persist within the review process. The primary limitation was the reliance on institutional subscriptions. As noted by Kitchenham and Charters (2007), excluding studies due to paywalls may lead to the omission of gray literature or articles from specific publishers, thereby limiting the comprehensiveness of the mapping study. Temporal Bias: Given the rapid evolution of security within CI/CD pipelines, studies published after the data collection period concluded were not included, posing a potential risk of technical obsolescence. A scarcity of scientific forums dedicated exclusively to the intersection of security and operations was identified, necessitating exhaustive manual searches to supplement the automated retrieval process. Furthermore, the uneven distribution of search results across different databases constitutes an external variable beyond the researchers' control. As highlighted by Petersen et al. (2015), this disparity stems from the distinct indexing policies and disciplinary scopes of each database (e.g., the predominantly theoretical focus of ACM compared to the applied focus of IEEE). Consequently, this variance could potentially influence the representational bias of certain software architectures over others.

7 Conclusions and future work

Embedding DevSecOps within the engineering process has emerged as a highly successful method for elevating both application caliber and defensive strength, largely by embedding safeguards directly into the initial phases of the production cycle. Embracing this methodology significantly curtails structural weaknesses, accelerates the discovery of potential hazards, and ensures strict adherence to cybersecurity mandates. Nevertheless, executing this model is not without obstacles, as companies frequently struggle with corporate pushback, a scarcity of expertly trained personnel, and substantial upfront financial investments.

The findings demonstrate that embracing the DevSecOps paradigm strongly promotes the inclusion of protective measures from the very beginning of the SDLC. This proactive stance ultimately fosters the development of highly robust and dependable digital infrastructures by prioritizing cross-team cooperation and seamless software integrations. Consequently, programming flaws are addressed immediately upon discovery, preventing them from evolving into convoluted operational nightmares. Under the DevSecOps philosophy, digital safety becomes a shared duty across all stakeholders. Collected literature consistently highlights that the core engine driving this methodology is the CI/CD pipeline, which seamlessly orchestrates the automation of programming tasks. Within this framework, choosing the most appropriate utilities becomes essential for executing rigorous defensive evaluations throughout the deployment chain. Furthermore, the analyzed data highlighted distinct hurdles impeding widespread DevSecOps acceptance, primarily tied to deeply rooted corporate mindsets, constrained financial or temporal resources, and inherent technological complexities. Ultimately, this review functions as an introductory map of current empirical studies regarding these modern practices, serving to spotlight existing advantages, pinpoint developmental gaps, and illuminate pathways for upcoming scholarly inquiries.

Projecting into the future, we foresee a significantly deeper fusion of artificial intelligence mechanisms within DevSecOps frameworks, alongside the formulation of comprehensive educational programs designed to smooth its transition across varied corporate environments. Such advancements will empower a broader spectrum of enterprises to successfully adopt this philosophy and fortify their digital creation workflows. Drawing from the comprehensive observations of this extended analysis, we pinpoint several paramount domains warranting further academic exploration:

- Next-Generation Defense for Cloud-Native Ecosystems: Upcoming academic efforts must prioritize the design of highly adaptable, scalable evaluation techniques capable of effectively auditing vast arrays of micro-containers without necessitating manual, one-by-one reviews.
- Harmonizing Shift-Left and Shift-Right Methodologies: Future inquires need to explore exactly how enterprises can optimally weigh proactive, pre-lease defensive evaluations against continuous, real-time operational monitoring and shielding.
- Artificial Intelligence and Machine Learning for Execution-Phase Protection: Leveraging intelligent algorithms to recognize behavioral irregularities and categorize potential attacks demonstrates immense potential; nonetheless, it demands far more rigorous scientific scrutiny.

Use of AI. AI tools were used for style and grammar correction, as well as specialized software for the visual mapping of literature. We declare that AI was not involved in the analysis of articles, data extraction, or the formulation of answers to the research questions. The authors assume full responsibility for the content, ensuring that no tools were used to generate or interpret the results presented.

References

Akbar, M. A., Khan, A. A., Mahmood, S., & Hyrynsalmi, S. (2025). Management of DevSecOps process: An empirical investigation. *Software: Practice and Experience, 55*(7), 1234–1255. https://doi.org/10.1002/spe.3419

Akbar, M. A., Rafi, S., Hyrynsalmi, S., & Khan, A. A. (2024). Towards people maturity for secure development and operations: A vision. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering* (pp. 528–533). Association for Computing Machinery. https://doi.org/10.1145/3661167.3661238

Alawneh, M., & Abbadi, I. M. (2022). Expanding DevSecOps practices and clarifying the concepts within Kubernetes ecosystem. In *2022 Ninth International Conference on Software Defined Systems (SDS)* (pp. 1–7). IEEE. https://doi.org/10.1109/SDS57574.2022.10062874

Ashenden, D., & Ollis, G. (2020). Putting the Sec in DevSecOps: Using social practice theory to improve secure software development. In *Proceedings of the New Security Paradigms Workshop 2020* (pp. 34–44). Association for Computing Machinery. https://doi.org/10.1145/3442167.3442178

Azad, N., & Hyrynsalmi, S. (2024). Multivocal literature review on DevOps critical success factors. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering* (pp. 520–527). Association for Computing Machinery. https://doi.org/10.1145/3661167.3661236

Barrak, A., Ksontini, E., Atike, R., & Jaafar, F. (2025). *FaaSGuard: Secure CI/CD for serverless applications—An OpenFaaS case study* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2509.04328

Billawa, P., Tukaram, A. B., Díaz Ferreyra, N. E., Steghöfer, J.-P., Scandariato, R., & Simhandl, G. (2022). SoK: Security of microservice applications: A practitioners' perspective on challenges and best practices. In *Proceedings of the 17th International Conference on Availability, Reliability and Security* (pp. 1–10). Association for Computing Machinery. https://doi.org/10.1145/3538969.3538986

Burkard, E. C. (2020). Usability testing within a DevSecOps environment. In *2020 Integrated Communications Navigation and Surveillance Conference (ICNS)* (pp. 1C1-1–1C1-7). IEEE. https://doi.org/10.1109/ICNS50378.2020.9222919

Cankar, M., Petrović, N., Pita Costa, J., Černivec, A., Antić, J., Martinčič, T., & Štepec, D. (2023). Security in DevSecOps: Applying tools and machine learning to verification and monitoring steps. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering* (pp. 201–205). Association for Computing Machinery. https://doi.org/10.1145/3578245.3584943

Cheenepalli, J., Hastings, J. D., Ahmed, K. M., & Fenner, C. R. (2025). Advancing DevSecOps in SMEs: Challenges and best practices for secure CI/CD pipelines. In *2025 13th International Symposium on Digital Forensics and Security (ISDFS)* (pp. 1–6). IEEE. https://doi.org/10.1109/ISDFS65363.2025.11011960

Chen, M., Liang, B., & Lu, X. (2024). The practice and application of a novel DevSecOps platform on security. In *2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)* (pp. 558–562). IEEE. https://doi.org/10.1109/AINIT61980.2024.10581700

Chen, T., & Suo, H. (2022). Design and practice of security architecture via DevSecOps technology. In *2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 310–313). IEEE. https://doi.org/10.1109/ICSESS54813.2022.9930212

Díaz, J., Pérez, J. E., Lopez-Peña, M. A., Mena, G. A., & Yagüe, A. (2019). Self-service cybersecurity monitoring as enabler for DevSecOps. *IEEE Access, 7*, 100283–100295. https://doi.org/10.1109/ACCESS.2019.2930000

Feio, C., Santos, N., Escravana, N., & Pacheco, B. (2024). An empirical study of DevSecOps focused on continuous security testing. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 610–617). IEEE. https://doi.org/10.1109/EuroSPW61312.2024.00074

Grigorieva, N. M., Petrenko, A. S., & Petrenko, S. A. (2024). Development of secure software based on the new DevSecOps technology. In *2024 Conference of Young Researchers in Electrical and Electronic Engineering (ElCon)* (pp. 158–161). IEEE. https://doi.org/10.1109/ElCon61730.2024.10468425

Haverinen, H., Janhunen, T., Päivärinta, T., Lempinen, S., Kaartinen, S., & Merilä, S. (2024). Automating cybersecurity compliance in DevSecOps with open information model for security as code. In *Proceedings of the 4th Eclipse Security, AI, Architecture and Modelling Conference on Data Space* (pp. 93–102). Association for Computing Machinery. https://doi.org/10.1145/3685651.3686700

Hermann, K., Peldszus, S., Steghöfer, J.-P., & Berger, T. (2025). An exploratory study on the engineering of security features. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)* (pp. 2470–2482). IEEE. https://doi.org/10.1109/ICSE55347.2025.00184

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering* (EBSE Technical Report EBSE-2007-01). Keele University.

Kwon, S., Son, W., & Lee, J.-H. (2025). Anomaly detection in containerized tactical systems using temporal graph neural networks. In *MILCOM 2025—IEEE Military Communications Conference* (pp. 1566–1571). IEEE. https://doi.org/10.1109/MILCOM64451.2025.11310523

Lazarus, J. I., Truett, L., Fischer, B., & Kershner, C. (2024). DevSecOps process assessment collaboration tool: A novel method to inject R&M into Agile development. In *2024 Annual Reliability and Maintainability Symposium (RAMS)* (pp. 1–5). IEEE. https://doi.org/10.1109/RAMS51492.2024.10457824

Le-Thanh, P., Le-Anh, T., & Le-Trung, Q. (2023). Research and development of a smart solution for runtime web application self-protection. In *Proceedings of the 12th International Symposium on Information and Communication Technology* (pp. 304–311). Association for Computing Machinery. https://doi.org/10.1145/3628797.3628901

Mahboob, J., & Coffman, J. (2021). A Kubernetes CI/CD pipeline with Asylo as a trusted execution environment abstraction framework. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 529–535). IEEE. https://doi.org/10.1109/CCWC51732.2021.9376148

Marandi, M., Bertia, A., & Silas, S. (2023). Implementing and automating security scanning to a DevSecOps CI/CD pipeline. In *2023 World Conference on Communication & Computing (WCONF)* (pp. 1–6). IEEE. https://doi.org/10.1109/WCONF58270.2023.10235015

Morales, J. A., & Yasar, H. (2023). Experiences with secure pipelines in highly regulated environments. In *Proceedings of the 18th International Conference on Availability, Reliability and Security* (Article 57, pp. 1–9). Association for Computing Machinery. https://doi.org/10.1145/3600160.3605466

Morales, J. A., Scanlon, T. P., Volkmann, A., Yankel, J., & Yasar, H. (2020). Security impacts of sub-optimal DevSecOps implementations in a highly regulated environment. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Article 63, pp. 1–8). Association for Computing Machinery. https://doi.org/10.1145/3407023.3409186

Moyón, F., Angermeir, F., & Mendez, D. (2024). Industrial challenges in secure continuous development. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice* (pp. 309–311). Association for Computing Machinery. https://doi.org/10.1145/3639477.3639736

Nadgowda, S., & Luan, L. (2021). Tapiserí: Blueprint to modernize DevSecOps for real world. In *Proceedings of the Seventh International Workshop on Container Technologies and Container Clouds* (pp. 13–18). Association for Computing Machinery. https://doi.org/10.1145/3493649.3493655

Nagasundari, S., Manja, P., Mathur, P., & Honnavalli, P. B. (2025). Extensive review of threat models for DevSecOps. *IEEE Access, 13*, 45252–45271. https://doi.org/10.1109/ACCESS.2025.3547932

Nugraha, A. D. P., & Irsan, M. (2025). Integrating self-protection into DevSecOps framework for defense against threat. In *2025 International Conference on Software Engineering and Computer Systems (ICSECS)* (pp. 287–291). IEEE. https://doi.org/10.1109/ICSECS65227.2025.11279258

Orosz, M., Spear, G., Duffy, B., & Charlton, C. (2022). Merging Agile/DevSecOps into the US DoD space acquisition environment—A multiple case study. *INSIGHT, 25*(4), 96–99. https://doi.org/10.1002/inst.12420

Patel, A., Laudya, R., Ragothaman, H., Udayakumar, S. K., Pandey, P., & Sheth, A. (2025). Dynamic secret injection for microservices in the cloud. In *2025 8th International Conference on Information and Computer Technologies (ICICT)* (pp. 72–79). IEEE. https://doi.org/10.1109/ICICT64582.2025.00018

Pecka, N., Ben Othmane, L., & Valani, A. (2022). Privilege escalation attack scenarios on the DevOps pipeline within a Kubernetes environment. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering* (pp. 45–49). Association for Computing Machinery. https://doi.org/10.1145/3529320.3529325

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology, 64*, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Pressman, R. S., & Maxim, B. R. (2021). *Ingeniería del software: Un enfoque práctico* (9.ª ed.). McGraw Hill.

Putra, A. M., & Kabetta, H. (2022). 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. https://doi.org/10.1109/ICOSNIKOM56551.2022.10034883

Rajapakse, R. N., Zahedi, M., & Babar, M. A. (2021). An empirical analysis of practitioners' perspectives on security tool integration into DevOps. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1–12). Association for Computing Machinery. https://doi.org/10.1145/3475716.3475776

Scanlon, T., & Morales, J. (2022). Revelations from an Agile and DevSecOps transformation in a large organization: An experiential case study. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering* (pp. 77–81). Association for Computing Machinery. https://doi.org/10.1145/3529320.3529329

Sermpezis, E., Karapiperis, D., & Tjortjis, C. (2024). Integration of security in the DevOps methodology. In *2024 15th International Conference on Information, Intelligence, Systems and Applications (IISA)* (pp. 1–6). IEEE. https://doi.org/10.1109/IISA62523.2024.10786669

Sinan, M., Shahin, M., & Gondal, I. (2025). Integrating security controls in DevSecOps: Challenges, solutions, and future research directions. *Journal of Software: Evolution and Process, 37*(6), Article e70029. https://doi.org/10.1002/smr.70029

Solaimalai, G. (2025). AI-augmented DevSecOps for cloud-native security automation. In *2025 International Conference on Recent Innovation in Science Engineering and Technology (ICRISET)* (pp. 1–8). IEEE. https://doi.org/10.1109/ICRISET64803.2025.11252281

Sommerville, I. (2011). *Software engineering* (9th ed.). Pearson.

Turner, R. (2021). Systems engineering and DevSecOps: Reviewing the principles. *INSIGHT, 24*(2), 38–43. https://doi.org/10.1002/inst.12339

Varela Gutiérrez, B. (2021). *Desarrollo seguro de software bajo la metodología DevSecOps*. Universitat Oberta de Catalunya. https://hdl.handle.net/10609/138449

Verderame, L., Caviglione, L., Carbone, R., & Merlo, A. (2023). SecCo: Automated services to secure containers in the DevOps paradigm. In *Proceedings of the 2023 International Conference on Research in Adaptive and Convergent Systems* (Article 10, pp. 1–6). Association for Computing Machinery. https://doi.org/10.1145/3599957.3606222

Wang, Z., Guo, G., Liu, C., & Zhu, W. (2022). Research on railway DevSecOps system construction based on "people-process-technology". In *2022 2nd International Signal Processing, Communications and Engineering Management Conference (ISPCEM)* (pp. 19–23). IEEE. https://doi.org/10.1109/ISPCEM57418.2022.00010

Yasar, H., & Teplov, S. E. (2022). DevSecOps in embedded systems: An empirical study of past literature. In *Proceedings of the 17th International Conference on Availability, Reliability and Security* (Article 155, pp. 1–6). Association for Computing Machinery. https://doi.org/10.1145/3538969.3544451

Yasar, H., Morales, J., Antunes, L., Earl, P., Edman, R., Hamed, J., Reynolds, D., Maffey, K. R., & Yankel, J. (2024). Insights on implementing a metrics baseline for post-deployment AI container monitoring. In *Proceedings of the 2024 International Conference on Software and Systems Processes* (pp. 46–55). Association for Computing Machinery. https://doi.org/10.1145/3666015.3666018

Yu, W., Qian, J., Xu, R., Jin, C., Fang, H., & Shi, X. (2024). Improving substation network security with DevSecOps and AIOps. In *2024 IEEE 10th Conference on Big Data Security on Cloud (BigDataSecurity)* (pp. 113–118). IEEE. https://doi.org/10.1109/BigDataSecurity62737.2024.00027

Yulianto, S., & Ngo, G. N. C. (2024). Enhancing DevSecOps pipelines with AI-driven threat detection and response. In *2024 International Conference on ICT for Smart Society (ICISS)* (pp. 1–8). IEEE. https://doi.org/10.1109/ICISS62896.2024.10751269

Zhang, H., Babar, M. A., & Tell, P. (2011). Identifying relevant studies in software engineering. *Information and Software Technology, 53*(6), 625–637. https://doi.org/10.1016/j.infsof.2010.12.010

Zhou, X., Mao, R., Zhang, H., Dai, Q., Huang, H., Shen, H., Li, J., & Rong, G. (2023). Revisit security in the era of DevOps: An evidence-based inquiry into DevSecOps industry. *IET Software, 17*(4), 435–454. https://doi.org/10.1049/sfw2.12132