Towards DevSecOps Model for Multi-tier Web Applications

Abderrahim Rida* and *Ayoub* Ait Lahcen

Engineering Sciences Laboratory, National School of Applied Sciences, Ibn Tofail University, Kenitra, Morocco

Abstract. Development, Security and Operations (DevSecOps) as part of Continuous Deployment (CD) or Continuous Delivery (CD) are considered a reliable link for communication and collaboration between development, security and operations teams. The philosophy of DevSecOps is defined as the ability to push new releases into production quickly and securely. We assume that DevSecOps brings new challenges, which primarily have a strong impact on decisions and organizational responsibilities within a company. We claim that PHP is the most commonly used back-end programming language for native applications, frameworks-based applications and content management system that implement this language. In addition, multi-tier architecture is one of the main architectures used by many software or web applications. Thereby, there is a significant and necessarily need to understand how DevSecOps on multi-tier PHP web applications could impact delivery accuracy. The goal of this work is to develop a DevSecOps pipeline model for multi-tier web applications that implement PHP.

1 Introduction

Organizations that implement IT architectures understand that the quality of software depends on the structure of the Continuous Integration/Continuous Delivery (CICD) pipeline [1].

DevSecOps process involves tasks carried out by software developers, IT professionals, and Cyber Security engineers to automate the release of software and infrastructure processes when commits are made. DevSecOps merges the responsibilities of teams to create a cohesive environment that ensures software excellence throughout its lifecycle. The main focus of DevSecOps is to enhance software quality by minimizing the time between commit (N) and commit (N+1), while ensuring that the software continues to perform well in the production environment [2].

It is crucial to encompass all software layers throughout their lifecycle, and multi-tier web applications need to go through a clearly defined CICD pipeline to enhance their performance with each new release. DevSecOps advocates for a proactive approach, incorporating key aspects such as building, compiling, testing, code analysing, security, and deployment with each new commit.

Multi-tier web applications consist of multiple layers, each necessitating its own security measures. Consequently, the challenge is evident, as ensuring robust security across all layers is a complex task [3, 4].

Additionally, the primary objective of a DevSecOps pipeline is to facilitate rapid delivery, even when numerous security protocols may hinder this swift process, Cyber Security engineers require additional time to assess the various stages and components of the pipeline, which creates an inherent issue between the speed-focused nature of DevOps and the need for thorough security measures, this situation results in a prioritization dilemma, where the assurance of either speed or security becomes a critical concern [3, 5]. Moreover, the integration of automation within the DevSecOps pipeline presents challenges, as certain steps in various stages are not easily automated, this limitation results in a slower pipeline process, as numerous tasks still necessitate manual configuration [3, 5]. On the same context, the absence of standardized configuration for DevSeOps pipelines represents a significant challenge for DevSeOps teams, leading to confusion and complicating the selection of appropriate tools that address all layers of the Software Development Life Cycle (SDLC) [3, 5], the reliance on open-source components introduces critical vulnerabilities, necessitating the establishment of well-defined policies to effectively mitigate the risks associated with these tools [5, 6]. Many organizations operate under a legal framework that governs their procedures, making the adoption of the DevSecOps model, which appears to fall outside this framework, a challenging endeavor [4, 7]. In the end, establishing effective collaboration among development, operations, and security teams is crucial yet challenging to achieve, this difficulty primarily stems from resistance exhibited by team members and a lack of adequate training and knowledge within the teams [3, 4, 5, 6].

Implementing DevSecOps in multi-tier architectures can pose various challenges, potentially impacting the software's overall quality. The DevSecOps pipeline must effectively balance speed and security when transitioning releases to the production environment. It

* Corresponding author: abderrahim.rida@uit.ac.ma

can be challenging to assume that the DevSecOps team will strictly adhere to a single pipeline guide, as some stages may be common while others are not.

In this paper, we present a proposal for a DevSecOps model tailored for multi-tier web applications. Existing literature predominantly emphasizes the DevSecOps pipeline concept, often overlooking comprehensive case studies.

The structure of the paper is outlined as follows: the second section offers background information that contextualizes the significance and relevance of DevSeOps. The third section reviews related literature, highlighting various state-of-the-art studies. The fourth section details the research design methodology employed. The fifth section articulates the anticipated contributions of the work. Finally, the concluding section discusses the implications and potential enhancements of the proposed DevSeOps pipeline model for multi-tier web applications.

2 Background

2.1 Multi-tier web applications

The multi-tier web application is structured with multiple layers, including Presentation (Client Tier), Application (Business Logic Tier), Data (Data Tier), and integration. While each layer is physically separated, they function together as a single unit with a logical aspect. Multi-tier architecture can be deployed as Microservices or a Monolithic platform. Regardless of the architecture chosen, each layer communicates with adjacent layers using specific communication protocols to send and receive requests.

On Basic multi-tier architecture, each layer sends requests and receives responses using protocols, this architecture contributes to the separation of responsibilities. As each layer performs a task, it takes inputs from the previous layer and sends outputs to the next and vice versa [8].

Understanding the dual nature of security in the context of delivery is crucial. The first aspect involves ensuring a secure pipeline process, which is established through collaboration among development, operations, and security teams. The second aspect pertains to the critical function of enhancing the integrity of inter-communication layers. Each layer must be fortified based on its specific characteristics and should effectively address various security challenges.

W3Techs conducted studies in which PHP was identified as the most popular server-side programming language (75.8% of usage), laborating a CICD pipeline for PHP multi-tier web applications would be a perfect idea. This will provide an accurate methodology for the DevSecOps team when handling multi-tier web applications developed by PHP.

Currently, PHP is supported by many hosting providers, as they offer a possibility to develop web applications from scratch or by using frameworks or CMS. PHP has become widespread because of [9]:

- Its compatibility with all major operating systems,
- Its high processing speed,
- Its large support community,
- Its flexibility to edit/add/delete according to need,
- Its Distribution of the output code with its license

2.2 From DevOps to DevSecOps

2.2.1 The evolution process of DevSecOps

The rapid delivery of software cannot be achieved without the integration of DevOps practices, prompting many organizations to prioritize both speed and security in response to new commits in their repositories. IT professionals recognize the significance of DevSecOps, viewing it as a critical framework for safeguarding against complex security threats that can impact the entire application stack. As an advancement of traditional DevOps, DevSecOps emphasizes the incorporation of security protocols throughout the software development process, reflecting a more mature approach to DevOps that integrates security considerations at every stage of the SDLC. This approach empowers DevSecOps teams to develop secure code and infrastructure while effectively addressing security vulnerabilities as they arise [10].

2.2.2 Pros and Cons of DevSecOps

The DevSecOps approach is increasingly being embraced by numerous organizations, driven by its advantages. However, while it introduces valuable features, it also presents certain challenges that necessitate heightened vigilance [11].

DevSecOps encompasses numerous advantageous characteristics that enhance communication and collaboration among development, operations, and security teams, thereby facilitating a more efficient and secure delivery process. By empowering teams to manage various stages of the SDLC, it enables timely identification of potential issues that could impede delivery. Additionally, this approach allows teams to respond more agilely to new customer requirements, ensuring optimal and accurate solutions are provided.

DevSecOps introduces challenges that may affect an organization's SDLC. Initially, it tends to emphasize speed rather than security due to the intricate nature of designing and modeling risk management vulnerabilities. Consequently, the security team requires additional time to comprehend the software's logic and implement appropriate security measures. Therefore, effective communication between the software development and security teams is essential to address all relevant aspects during the pipeline execution.

2.2.3 DevSecops challenges and solutions

Numerous studies address the challenges that DevSecOps may encounter, while also proposing corresponding solutions to these problems [12].

These challenges can be categorized into four primary themes. The first theme, People, encompasses issues related to knowledge, skills, collaboration, and the mindset of team members. The second theme,

Practices, addresses the transition from manual to automated processes, the management of continuous and rapid security assessments throughout the SDLC, and the understanding of the gap between security and DevSecOps due to the pace of change. The third theme, Tools, involves challenges associated with tool selection, integration, configuration, and vulnerabilities. The fourth theme, Infrastructure, pertains to the complexities of operating within resource-constrained and highly regulated environments.

Addressing these challenges necessitates a variety of solutions across different domains. People: it is essential to standardize communication and collaboration procedures among teams, clearly delineate team responsibilities, and foster strong engagement in shared tasks. Practices: the implementation of standardized mechanisms, policies, and models within CI/CD frameworks is crucial, alongside the automation of vulnerability detection and the prioritization of security through continuous measures. Tools: similar standards and automation strategies should be employed to enhance security prioritization and ongoing protective actions. Infrastructure: testing the pipeline within a distinct environment, implementing stringent access and management policies, and enhancing DevSecOps by establishing a supportive framework.

3 Related work

By implementing strategies across the whole SDLC that ensure quick and secured delivery, DevSecOps significance is really clear. Static Code Analysis, Dynamic Application Security Testing, Container Security, and Penetration Testing make up the security-focused portion of the DevOps pipeline. Consideration of infrastructure as a component of the DevSecOps pipeline is required to attain a reduction of the entire attack surface. To improve security analytics, AI and machine learning technologies could be incorporated into DevSecOps procedures [2].

The Microservices DevSecOps pipeline could be viewed as a framework consisting of Container Security, Secrets Management, Vulnerability Scanning, Access Control, Secure Deployment Practices, Continuous Monitoring and Logging. This pipeline includes Code and Infrastructure to achieve the sacred goal: integration security measures along the SDLC [13].

In the context of DevSecOps Pipeline [14], a number of tools, including Web Application Firewalls (WAF), Runtime Application Self-Protection (RASP), Interactive Application Security Testing (IAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA), should be implemented as part of the automation strategy to safeguard the software throughout its Life Cycle. Regardless of the CICD tool, an unreliable pipeline will undoubtedly result in subpar application development. Thus, in order to attain a high-security pipeline level, it would be beneficial to adhere to a few security tasks. These tasks include:

• Making sure that servers hosting code and artifact repositories are hardened,

• Safeguarding the credentials for gaining access to repositories,

• Logging all code and building update activities

DevSecOps challenges related to tools, practices, infrastructure and people are well defined and addressed through proposed solutions [15]. It was not possible to cover all DevSecOPs challenges at once as updates to methodologies, technologies and practices are being made. This led to the discovery that there is a gap area that should be taken into account by researchers. Thereby, some aspects can be identified that need to be investigated to ensure a high quality DevSecOPs pipeline:

• Security Testing Tools and Methodologies,

• Evaluating Security Tools,

• Balancing Security and Speed of Softwar Delivery,

• Introducing Security Metrics in DevSecOps Practices and Establishing Guidelines for the DevSecOps Pipeline,

• Considering Cultural or Human Issues,

• Considering Resource-constrained and highly regulated infrastructures,

• Implementation of shift-left security and continuous security

The complexity of the DevSecOps pipeline has a socio-technical aspect [16], as operating a secure CICD workflow comes with a personal and organizational mindset that defines the rapid development, deployment and operation of software and software-based systems. This leads to this pipeline flow should move the security of the following processes as part of each individual process such as: Plan, Develop, Build, Test, Release, Deploy, Operate, Monitor and Feedback. Security practices that fluctuate frequently need to be implemented by DevSecOps teams at every stage and step.

Application security testing should be integrated into the CI/CD pipeline. We must manage [17]: Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Vulnerability Scanning and Penetration, Configuration Management, Infrastructure as Code and Continuous Monitoring. Additionally, Security Automation, Leftward Shift of Security, and Leftward Cost and Value management are the key benefits of DevSecOps practice. Although good collaboration and an understanding of security practices could be shared between teams, there are organizational, technical, operational and personal challenges to the reliability of the DevSecops pipeline.

The DevSecOps workflow potentially represents a large attack surface from a security perspective, and any improvement in the way security is designed in the build system itself has a huge impact on the security of the end product. The secure CICD pipeline is typically heterogeneous and includes various subsystems, which are quite complex in themselves, as it is challenging to make the security of this pipeline hermetic [18].

The integration of security measures within the DevSecOps pipeline enhances the overall security posture of the software development lifecycle. By incorporating tools such as WAF, RASP, IAST, DAST, and SCA, organizations can proactively identify and mitigate vulnerabilities, thereby reducing the attack

surface. Additionally, the emphasis on hardening servers, safeguarding credentials, and logging activities fosters a culture of accountability and transparency, which is crucial for maintaining the integrity of the software delivery process. The socio-technical aspect of DevSecOps also encourages collaboration and shared responsibility among teams, leading to faster and more secure deployments.

Despite its advantages, the complexity of the DevSecOps pipeline can create challenges in implementation. The heterogeneous nature of the pipeline may lead to difficulties in achieving seamless security integration across various subsystems. Moreover, the ongoing evolution of methodologies and technologies means that organizations may struggle to keep up with best practices, potentially leaving gaps in security. Additionally, the need for a cultural shift within organizations can meet resistance, as teams may be reluctant to adopt new processes or share responsibility for security, ultimately hindering the effectiveness of the DevSecOps approach.

4 Research design

This research is based on an empirical study on implementing a secure CI-CD pipeline in multi-tier PHP web applications. Our main research questions (RQ) are as follows:

• **RQ1**. Is there any guideline or reference pipeline architecture to follow for implementing secured CICD on PHP multi-tiers web application?

• **RQ2**. Does this reference guide guaranty any accurate production release?

• **RQ3**. Could this DevSecOps pipeline be implemented easily on software that implement other programming language?

4.1 Systematic literature review

We adopt Systematic Literature Review (SLR), as it's known one of the widely used research methods in Evidence-Based Software Engineering (EBSE) paradigm [18]. Its goal is providing a well-defined process for identifying, evaluating, and interpreting all available topics relevant to a DevSecOps on PHP multi-tier web applications in order to establish solid background knowledge.

4.2 Exploratory case study

An empirical study should be conducted using a specific research method based on the nature of the topic under study and the research questions to be answered [19]. There is already a bit of research on implementing DevSecOps in PHP multi-tier web applications. This is why we decided to conduct an exploratory case study. Case studies are considered a good research method to conduct research in a real-world context. Since this research has broad and high-level goals and there is little empirically collected knowledge about the DevSecOps pipeline for multi-tier PHP web applications, we focus on a real-world case when conducting it.

4.3 Data collection

We will collect information [20] from documentation (published electronic sources, general websites) based on qualitative and quantitative data. This secondary data method research is an essential component that can help obtain information from previous studies as a basis for conducting a research or as necessary background information. It can also help design a study and provide a basis for comparing the study's primary results.

4.4 Data analysis

The collected data will be examined using thematic analysis method [21].We will analyze the data to suggest a DevSecOps model for PHP multi-tier web applications. We assume to put this architecture on challenge for allowing enterprises who used PHP as a back-end programming language to implement this DevSecOps Model. The aim goal of this technique is to check the accuracy level of this DevSecOps model.

5 Our approach

Having a global idea from related work leads us to think about making a concrete DevSecOps Pipeline, which will be a challenge. As we have already seen, PHP is the most commonly used language, which leads us to think about a reference architecture of a secured CICD pipeline for PHP multi-tier web application. Our expected post will focus on implementing DevSecOps reference architecture that intend to guarantee fast and secure deliver.

To show an overall picture of the infrastructure chosen, let's discuss how it was created, the choice of Amazon Web Services (AWS) as a top popular cloud provider, was defined using W3Techs. Additionally, given its many benefits, we planned to use Infrastructure as Code (IaC) as It allows Automation of infrastructure management, Effectiveness in resource allocation, Time saving and reduction, Speed and agility of deployment, Reducing human error and manual errors, Flexibility for the backup and duplication environment and reliability for secure CICD.

We assume that a multi-tier PHP web application using a specific CMS, has been developed for this research purpose. Using AWS, this application was deployed in a real production environment with the services It needs.

Our goal is to keep the real production environment working properly. Moreover, making sure a new commit is functioning properly and has no effect on the performance of the production environment at this time is the objective. To do this, we intend to launch three EC2 instances (Development instance, Production instance and Jenkins instance) so that we can apply the DevSecOps pipeline model and ensure that, prior to deploying to the production environment, every release doesn’t result in regressions. Thereby and taking account of IAC, we prepared a Terraform file that generate the desired EC2 instances, which results in the followed proposal pipeline.

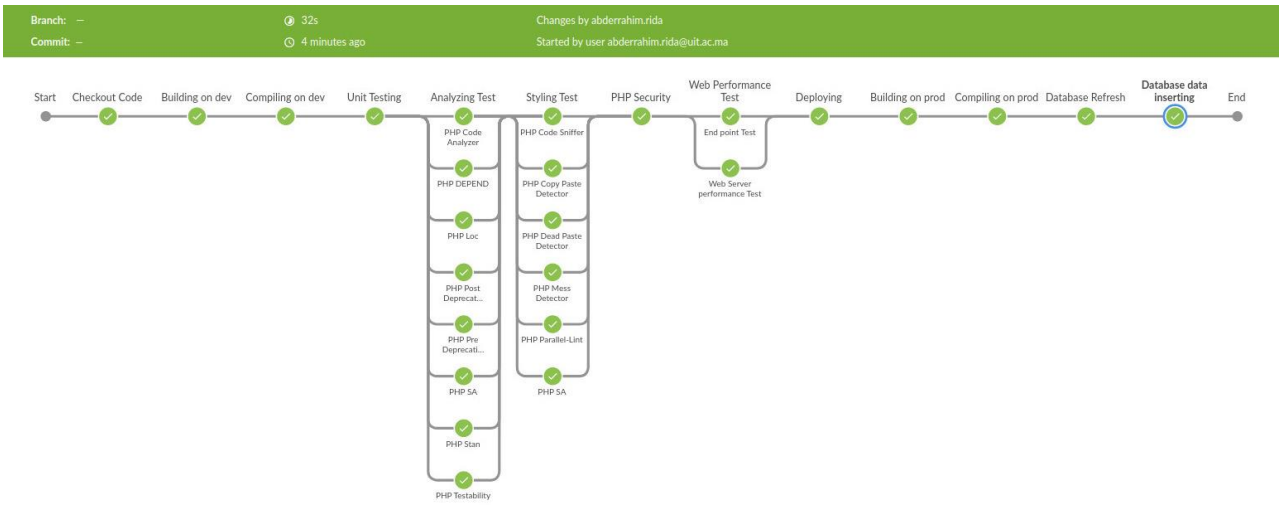

Fig. 1. Proposed DevSecOps Pipeline

Although our DevSecOps Model Pipeline is accurate, we are aware that updates and adjustments are required. As a result, we plan to bring best practices that will assist DevSecOps teams in setting up their pipelines on multi-tier web applications created with PHP or other backend programming languages, so they can begin designing a good pipeline that will ensure secure and quick delivery. In other words, the above figure displays a version of the DevSecOps Pipeline model that will be refined at the conclusion of this study.Automation is implemented at every stage to facilitate the continuous integration and continuous delivery (CI/CD) process. Let's quickly go over what each step of the process might entail.

• Start: The initiation of the pipeline occurs when events, such as a new commit, activate the workflow,

• Checkout Code: The current task involves obtaining the most recent commit from the GitLab repository,

• Building on Dev: This phase is intended to verify that the processes of resolving dependencies, generating binaries, and executing build operations are clearly defined within the development environment,

• Compiling on Dev: The compilation process ensures the correctness of the code's syntax, which involves verifying the absence of errors in the PHP code,

• Unit Testing: This phase, referred to as Unit Testing, encompasses three types of testing: Unit Testing, Integration Testing, and Functional Testing, facilitated by the capabilities of the employed library,

• Analyzing Test: At this stage, the assessment of code quality is conducted, including a review of PHP coding standards, measurement of PHP code complexity, and identification of potential design flaws,

• Styling Test: This section emphasizes the importance of compliance with established coding style guidelines,

• Security Test: This Section, characterized by many security libraries and patches, focuses on the security assessment of PHP code. It aims to identify vulnerabilities and confirm adherence to security best practices. The primary objective is to guarantee that the code is free from potential security risks,

• Web Performance Test: A Web Performance Test evaluates the operational capabilities of a web server. It determines whether API endpoints and web pages function as intended, while also assessing server performance indicators like response time and throughput. This process confirms that the server can manage expected traffic volumes efficiently.

• Deploying: A new commit is currently being pulled from the GitLab repository,

• Building on Prod: This stage mirrors the Building on Dev stage, focusing specifically on the production environment to guarantee that the build meets production standards,

• Compiling on Prod: This process involves compiling the code that is ready for production, ensuring it is optimized and suitable for deployment in a live setting.

• Database Refresh: Make database schema updated or refreshed to guarantee that the latest structure and records are used in the production environment,

• Database data inserting: The goal's section is to insert necessary or initial data into the production database,

• End: This stage shows the end of the pipeline, with the application delivered

6 Conclusion

This paper describes the DevSecOps pipeline model for multi-tiered PHP web application hosted on Cloud using AWS.This model will be perfected step by step along our study. It will be a reference guideline for any DevSecOps team that handle multi-tier web application developed by PHP or by other back-end programming languages. The aim goal is to preserve the real production environment for any potential risk or regression. The research is ongoing, driven by the objective of developing a reference DevSecOps pipeline tailored for multi-tier web applications. The current trajectory emphasizes the necessity of integrating insights from existing literature to formulate a robust pipeline proposal, which must be regularly revised to incorporate emerging technologies. Additionally, obtaining feedback from DevSecOps teams across various organizations presents a valuable opportunity to

refine the proposed model, enhancing its accuracy and overall effectiveness.

References

1. B. Jambunathan and D. Y. Kalpana, "Design of DevOps solution for managing multi-cloud distributed environment, " *Int. J. Eng. Technol*., **7**, no. 33, pp. 637-641, (2018).
2. O. O. Abiona, J. Oladapo, N. Oluwole, O. C. Oyeniran, A. Okechukwu, and N. Abiola, "The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline," *World J. Adv. Eng. Technol. Sci.*, **11**, no. 2, pp. 127–133, (2024). doi: 10.30574/wjaets.2024.11.2.0093.
3. J. Kuruvilla, "8 Common DevSecOps Challenges and How to Overcome Them," *Adaptavist*, Dec. 13, 2021. [Online]. Available: https://www.adaptavist.com/blog/8-common-devsecops-challenges-and-how-to-overcome-them.
4. V. Prasad, "Building a Strong DevSecOps Culture in a Hybrid and Multi-Cloud Environment," *AppSec Engineer*, updated Dec. 26, 2023. [Online]. Available: https://www.appsecengineer.com/blog/building-a-strong-devsecops-culture-in-a-hybrid-and-multi-cloud-environment.
5. R. Badam and Team Qentelli, "DevSecOps Challenges at Scale," *Qentelli*, [Online]. Available: https://qentelli.com/thought-leadership/insights/devsecops-challenges-scale.
6. D. H. A. Gonçalves, *DevSecOps for web applications: a case study*, Master's Thesis, Instituto Superior de Engenharia do Porto, (June 2022).
7. A. Kharchuk and O. Boyko, "Web Application Development Challenges," *SPD Tech*, Sep. 9, 2023. [Online]. Available: https://spd.tech/web-development/web-application-development-challenges/.
8. C. Cerbulescu and M. Cerbulescu, "N-Tier applications: XML-XSLT in data traffic optimization," presented at the *7th Int. Conf. Dev. Appl. Syst.*, Suceava, Romania, May 27–29, 2004, pp. 423–425.
9. S. Sotnik, V. Manakov, and V. Lyashenko, "pReview: Overview: PHP and MySQL features for creating modern web projects," *Int. J. Acad. Inf. Syst. Res. (IJAISR)*, Jan. 2023. [Online]. Available: https://openarchive.nure.ua/server/api/core/bitstreams/99c2a8e9-aefd-47d4-a06f-838650b760db/content.
10. E. Katz, "What is the DevSecOps Maturity Model (DSOMM)?," *SpectralOps*, Feb. 15, 2024. [Online]. Available: https://spectralops.io/blog/what-is-the-devsecops-maturity-model-dsomm/.
11. Veritis, "Pros and Cons of DevSecOps," Jul. 2022. [Online]. Available: https://veritis.com/wp-content/uploads/2022/07/Pros-and-Cons-of-DevSecOps.pdf.
12. R. N. Rajapakse, M. Zahedi, M. Ali Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: A systematic review," 2021. [PDF].
13. A. Singh and A. Aggarwal, "Securing microservice CI/CD pipelines in cloud deployments through Infrastructure as Code implementation approach and best practices," *J. Sci. Technol.*, vol. 3, no. 3, pp. 51-65, May 2022. [Online]. Available: https://thesciencebrigade.com/jst/article/view/203.
14. O. B. Abiola and O. G. Olufemi, "An enhanced CI/CD pipeline: A DevSecOps approach," *Int. J. Comput. Appl.*, vol. 184, no. 48, pp. 8-13, Feb. 2023. doi: 10.5120/ijca2023922594.
15. R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: A systematic review," *Inf. Softw. Technol.*, vol. 141, p. 106700, 2022. doi: 10.1016/j.infsof.2021.106700.
16. C. Woody, T. Chick, A. Reffett, S. Pavetti, R. Laughlin, B. Frye, and M. Bandor, DevSecOps pipeline for complex software intensive systems: Addressing the cybersecurity challenges, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2020.
17. B. Jammeh, "DevSecOps: Security expertise a key to automated testing in CI/CD pipeline," Bournemouth Univ., 2020. [Online]. Available: https://www.researchgate.net/publication/347441415.
18. V. Dakic, J. Redzepagic, and M. Basic, "CI/CD toolset security," in *Proc. 33rd DAAAM Int. Symp. Intell. Manuf. Autom.*, Vienna, Austria, Oct. 27–28, 2022, pp. 161–164.
19. B. A. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep., 2007.
20. S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. K. Sjøberg, Eds., London, U.K.: Springer, 2008, pp. 285–311.
21. E. Paradis, B. O'Brien, L. Nimmon, G. Bandiera, and M. A. Martimianakis, "Design: Selection of data collection methods," *J. Grad. Med. Educ.*, vol. 8, no. 2, pp. 263–264, May 2016. doi: 10.4300/JGME-D-16-00098.1. E. Paradis, B. O'Brien, L. Nimmon, G. Bandiera, and M. A. Martimianakis, "Design: Selection of data collection methods," *J. Grad. Med. Educ.*, vol. 8, no. 2, pp. 263–264, May 2016. doi: 10.4300/JGME-D-16-00098.1.
22. V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qual. Res. Psychol.*, vol. 3, no. 2, pp. 77–101, 2006.