Software Engineering in Computing Systems

E-ISSN: 3124-0062

*Research Article*

# Framework for Integrating Continuous Integration and Continuous Deployment (CI or CD) with Automated Security Testing to Improve Software Dependability

**Syaiful Anwar [1]*, Irwanto [2], Safrizal [3]**

[1] Institut Teknologi dan Bisnis Dewantara; e-mail : syaifulanwar101@yahoo.com
[2] Universitas Patimura; e-mail : irwanto@lecturer.unpatti.ac.id
[3] Universitas Pembangunan Jaya ; e-mail : safrizal.abdurrahman@upj.ac.id
* Corresponding Author : Syaiful Anwar

**Abstract:** The increasing demand for rapid software delivery has led to the widespread adoption of Continuous Integration (CI) and Continuous Deployment (CD) pipelines. These pipelines automate the processes of code integration, testing, and deployment, significantly improving the speed and reliability of software development. However, traditional CI or CD pipelines often overlook security testing, leading to vulnerabilities in the deployed software. To address this gap, this study proposes an integrated framework that embeds automated security testing within the CI or CD process. The framework incorporates security testing tools such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Vulnerability Assessment and Penetration Testing (VAPT) to ensure continuous security checks throughout the development lifecycle. The experimental results show that the proposed framework enhances early vulnerability detection, with detection rates increasing from 30% to 70%. Additionally, the framework reduces deployment failures from 50% to 20%, demonstrating its effectiveness in improving software dependability. While the integration of automated security testing adds a slight 5% increase in pipeline execution time, this minimal impact does not significantly affect the overall speed of the pipeline. The proposed approach successfully balances security and efficiency, ensuring that software is both secure and delivered at high speed. This research highlights the importance of integrating security into CI or CD pipelines and demonstrates that it is possible to achieve high security without sacrificing the speed of software development. The study also discusses the practical implications for software development teams and suggests areas for future research, including the integration of advanced AI-driven security testing tools and the expansion of the framework's applicability across different software projects.

**Keywords:** Automated Security; Continuous Deployment; Continuous Integration; Software Dependability; Vulnerability Detection.

**Received: November 20, 2025**
**Revised: Desember 30, 2025**
**Accepted: January 14, 2026**
**Published: January 20, 2026**
**Curr. Ver.: January 20, 2026**

## 1. Introduction

In the rapidly evolving world of software development, the need to deliver high-quality products at an unprecedented pace has become a critical requirement. Traditional software development methodologies, which are often slow and cumbersome, have struggled to meet the demands of modern businesses and consumers. This has led to the widespread adoption of more agile, automated, and efficient approaches such as Continuous Integration (CI) and Continuous Deployment (CD) [1]. The increasing demand for faster delivery cycles has necessitated the need for robust and scalable software delivery pipelines. However, while CI or CD pipelines have significantly accelerated the development and deployment processes, they often fail to integrate systematic security testing, resulting in vulnerabilities in deployed systems [2].

The CI or CD pipeline facilitates the integration of code changes and their deployment into production environments, enabling faster feedback loops, earlier bug detection, and more efficient releases. Continuous Integration involves the regular merging of code changes from multiple developers into a shared repository, ensuring early detection of integration issues and maintaining the stability of the codebase [3]. Similarly, Continuous Deployment extends this practice by automating the process of deploying code changes directly into production environments, thereby enhancing the speed and reliability of software updates [4]. While these pipelines have streamlined many aspects of software development, they often overlook the integration of automated security testing, which is crucial in ensuring that vulnerabilities are detected early in the development lifecycle [5].

Agile methodologies, which emphasize iterative development and regular releases, align closely with the goals of CI or CD, providing a framework for continuous improvements and stability [6]. CI or CD practices, particularly when integrated with automated security testing, can address the challenges faced by traditional software testing methods, which are often reactive rather than proactive. The integration of security testing into CI or CD pipelines can help identify vulnerabilities early, thus reducing the likelihood of security breaches in production environments [7].

In this paper, we propose an integrated framework that embeds automated security testing within the CI or CD pipeline to enhance software dependability. This framework aims to address the gap between rapid software delivery cycles and the need for rigorous security testing by automating vulnerability detection within the CI or CD workflow [8]. By incorporating automated security testing tools into the pipeline, the framework ensures that vulnerabilities are detected early without significantly increasing the execution time of the pipeline. Through experimental validation, the framework's effectiveness in improving vulnerability detection and reducing deployment failures will be assessed.

In the evolving landscape of software development, the rapid delivery of high-quality products has become increasingly important. Continuous Integration (CI) and Continuous Deployment (CD) pipelines have emerged as key enablers of this swift delivery, automating the process of code integration, testing, and deployment. These pipelines, while crucial for enhancing development speed and efficiency, often overlook the systematic integration of security testing, leaving software systems vulnerable to various threats [5]. The failure to embed security testing early in the development cycle can lead to severe vulnerabilities, jeopardizing the software's integrity, confidentiality, and availability.

CI or CD pipelines primarily focus on streamlining the build and deployment processes, ensuring that software updates are delivered rapidly and frequently. However, this efficiency comes at the expense of security integration, which is frequently addressed late in the development cycle. This delayed approach to security, commonly known as "shifting security left," is inadequate for the demands of modern software development environments that require continuous and proactive security assessments [9]. Furthermore, the manual management of security components in traditional CI or CD pipelines is labor-intensive and prone to errors, particularly as the number of services and applications increases [10]. These challenges highlight the need for a more integrated and automated approach to security within CI or CD pipelines.

To address these concerns, we propose an integrated framework that embeds automated security testing directly within the CI or CD process. The goal is to enhance software dependability by making security a continuous and seamless part of the development lifecycle. This framework incorporates various automated security checks, including Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Vulnerability Assessment and Penetration Testing (VAPT) tools, to ensure continuous security evaluation without disrupting the development flow [11]. Additionally, the integration of Artificial Intelligence (AI) and Machine Learning (ML) technologies for anomaly detection and predictive analytics promises to further bolster the security of software systems by identifying and mitigating threats in real time [12].

The proposed framework not only aims to automate security testing but also to encourage collaboration between development, operations, and security teams-a practice known as DevSecOps [13]. By embedding security checks throughout the CI or CD pipeline, the framework ensures that vulnerabilities are detected and addressed early, reducing the likelihood of deploying insecure software and improving overall software dependability.

## 2. Literature Review

### CI or CD Pipelines: Evolution, Benefits, and Challenges in the Software Development Lifecycle

The rapid pace of software development demands highly efficient and automated practices, which has led to the widespread adoption of Continuous Integration (CI) and Continuous Deployment (CD) pipelines. CI or CD pipelines aim to streamline the software delivery process by automating the build, test, and deployment stages, thus ensuring faster and more reliable software releases [14]. Initially, CI focused on the continuous integration of code from different developers into a shared repository, allowing for earlier detection of integration issues and ensuring software stability. CD, which extends CI by automating the deployment of software into production, ensures that new features and updates are delivered quickly and reliably to end-users [15].

Over time, the scope of CI or CD pipelines has expanded beyond software to include Cyber-Physical Systems (CPS), where the integration of hardware and software expertise within CI or CD pipelines ensures high dependability despite the unique challenges of combining these two domains [16]. Furthermore, the integration of AI-driven solutions has revolutionized CI or CD pipelines, improving automation, scalability, and efficiency. AI-powered tools enhance the decision-making process by providing predictive analytics and anomaly detection, which help mitigate potential issues before they become problematic [17].

Despite the undeniable benefits of CI or CD, the adoption of these pipelines presents several challenges. Cultural shifts within organizations are required, as CI or CD necessitates significant changes to traditional workflows, communication, and collaboration among teams [18]. Moreover, the lack of interoperability between various CI or CD tools, which often support different programming languages and platforms, poses a major obstacle to seamless integration across the pipeline [5]. Another key challenge is ensuring that security is adequately addressed within CI or CD pipelines, which often prioritize speed and efficiency over security considerations.

Digital technology development has encouraged organizations to adopt more adaptive and automated approaches to software development. One of the widely used approaches is Continuous Integration and Continuous Deployment (CI/CD), which enables the processes of code integration, testing, and software distribution to be carried out continuously and in a structured manner. This approach helps development teams accelerate the software development cycle while improving system quality through consistent testing processes.

In the context of digital transformation, the integration of various technologies such as machine learning, blockchain, and modern computing systems indicates that integrated system development requires a framework capable of supporting continuous process automation and system security [19]. The implementation of an adaptive technological framework allows information systems to evolve dynamically in accordance with organizational needs and increasingly complex digital environments.

Furthermore, the integration of intelligent technologies into digital systems also demonstrates that modern software development is not only focused on application functionality but also on information technology–based innovation management processes that support operational efficiency and the sustainability of digital systems [20]. Therefore, CI/CD has become an important approach in establishing a sustainable software development process integrated with security technologies and system automation.

### Security Testing in CI or CD: Analysis of Existing Practices, Common Pitfalls, and Why Security Becomes an Afterthought

While CI or CD pipelines offer numerous advantages, security often becomes an afterthought. The rapid pace of CI or CD practices, with their focus on speed and continuous delivery, can lead to a deprioritization of security, leaving software vulnerable to potential attacks. Security testing is frequently incorporated at the end of the development cycle, which is insufficient for the fast-paced, iterative nature of CI or CD environments [9]. As a result, vulnerabilities often go undetected until after deployment, when they are more difficult and costly to address.

To mitigate these risks, automated security testing techniques, such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Vulnerability Assessment and Penetration Testing (VAPT), have been integrated into CI or CD pipelines. These tools enable continuous security assessments throughout the development lifecycle, helping to identify vulnerabilities early and ensuring that security is consistently maintained [11]. Additionally, the DevSecOps approach, which integrates security practices into every stage of the CI or CD pipeline, ensures that security is not an afterthought but an integral part of the development process [21].

However, despite the integration of these practices, several common pitfalls persist. One of the most significant issues is the late integration of security testing, which still occurs in many CI or CD workflows. This late integration creates gaps where vulnerabilities can be missed, increasing the chances of deploying insecure software [9]. Security misconfigurations are another common pitfall, with many CI or CD pipelines suffering from misconfigurations that leave systems open to exploitation. Studies indicate that fewer than 1% of CI or CD workflows are free from such misconfigurations [22]. Furthermore, security testing is often overlooked or inadequately implemented due to a lack of awareness or expertise in security practices among development teams [18].

The focus on speed, combined with the complexity of integrating security into the CI or CD pipeline, leads to security being neglected. Organizations may not have the resources or expertise to implement robust security measures, and as a result, vulnerabilities are left unaddressed [14]. Furthermore, security becomes an afterthought due to the constant pressure to deliver software quickly, often at the expense of rigorous testing.

### Automated Security Testing: Role and Effectiveness of Automated Security Testing Tools

Automated security testing has become a fundamental aspect of modern software development, particularly within Continuous Integration (CI) and Continuous Deployment (CD) pipelines. These tools, including Static Application Security Testing (SAST), are designed to identify vulnerabilities early in the development process, ensuring that issues such as memory-related vulnerabilities and other security risks are detected before deployment [23]. The adoption of automated security tools is especially crucial in preventing vulnerabilities that could otherwise compromise the security, integrity, and performance of software systems.

Despite their importance, automated security testing tools face several challenges. One of the most significant hurdles is the high rate of false positives, where the tools incorrectly flag non-issues as vulnerabilities, leading to unnecessary remediation efforts [24]. Additionally, usability issues often arise, as developers struggle to interpret and act on the warnings generated by these tools. This can lead to vulnerabilities being ignored or poorly addressed, exacerbating security risks in the final product [25]. Furthermore, the integration of security testing tools into the development workflow can be challenging, as it requires seamless collaboration with issue-tracking systems and other development tools [23]. This integration is crucial for streamlining the identification, communication, and resolution of security vulnerabilities, thus making the remediation process more efficient.

To address these challenges, advanced techniques such as fuzz testing and the application of genetic algorithms for feature selection in vulnerability detection tools are being explored. Fuzz testing involves automatically generating a large number of test inputs to detect vulnerabilities in software systems, particularly those related to unexpected user inputs [24]. The use of genetic algorithms further enhances the accuracy and efficiency of automated security testing by optimizing the selection of test features, ultimately improving the detection rate of vulnerabilities [26]. Combining multiple automated testing methods, including static and dynamic testing, can significantly enhance the overall effectiveness of security testing in CI or CD pipelines [27].

### Dependability in Software: Concepts and Importance in CI or CD Pipelines

Dependability is a critical attribute for any software system, encompassing reliability, availability, maintainability, and safety. In the context of CI or CD pipelines, dependability is essential for ensuring that software updates do not introduce new faults and that any issues are quickly identified and resolved [28]. Reliability refers to the software's ability to consistently provide correct service, while availability ensures that the system is ready to operate whenever required. Maintainability involves the ease with which software can be updated and modified, and safety relates to the absence of catastrophic consequences resulting from software failures [27].

Achieving dependability in software requires comprehensive fault management strategies, including fault avoidance, fault tolerance, fault removal, and fault forecasting. These strategies aim to mitigate the impact of errors and failures, ensuring that the software continues to function correctly even in the presence of issues [25]. In CI or CD pipelines, dependability is especially important because the continuous nature of software integration and deployment can introduce unforeseen problems, particularly when security vulnerabilities are overlooked. Automated testing, monitoring, and issue management practices are essential for maintaining dependability in CI or CD environments by identifying potential faults early in the development process [29].

Moreover, dependability can be measured and evaluated through various methods, including the use of software tools that model and visualize reliability, availability, and maintainability metrics. These evaluations provide valuable insights for improving the dependability of software systems by comparing different approaches and identifying areas for improvement [28]. The integration of automated security testing tools within CI or CD pipelines plays a significant role in enhancing dependability by ensuring that security vulnerabilities are detected and resolved early, thus minimizing the risk of failures during deployment.

## 3. Research Method

The research employs a Design Science Research (DSR) approach to develop a framework that integrates automated security testing into Continuous Integration and Continuous Deployment (CI or CD) pipelines, addressing the lack of systematic security testing in traditional CI or CD workflows. The framework incorporates tools like Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Vulnerability Assessment and Penetration Testing (VAPT) to detect vulnerabilities early without hindering the development process. By integrating these tools into CI or CD platforms such as Jenkins and GitLab, the framework ensures continuous security assessments throughout the software lifecycle. The framework's effectiveness is validated through an experimental setup simulating a real-world CI or CD environment, demonstrating improved vulnerability detection and enhanced security without significantly affecting deployment speed. This approach provides a practical solution for maintaining software security and dependability in fast-paced development environments.

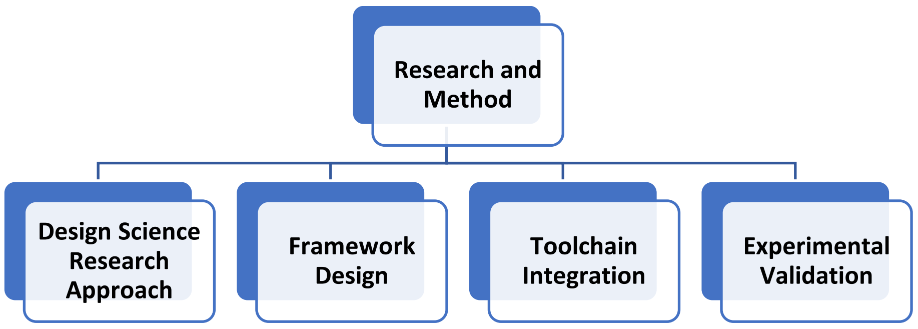

**Figure 1.** Flowchart structure.

### Design Science Research Approach

The research methodology employed in this study follows the Design Science Research (DSR) approach, which is commonly used to develop and evaluate artifacts that address practical issues. DSR focuses on the creation and iterative refinement of models, frameworks, and methodologies to solve identified problems. In this study, the primary problem is the lack of systematic integration of security testing in traditional CI or CD pipelines, leading to vulnerabilities in deployed software systems.

The DSR approach involves multiple cycles of design, testing, and improvement to ensure that the proposed framework effectively integrates automated security testing within the CI or CD pipeline. This iterative process guarantees that the framework is both practical and relevant to real-world software development environments, where rapid software releases must be balanced with security considerations. By leveraging DSR, the study ensures the development of a framework that directly addresses the gaps in traditional CI or CD practices that overlook security.

## Framework Design

The proposed framework integrates automated security testing tools within the CI or CD pipeline to ensure continuous security assessments throughout the software development lifecycle. The framework incorporates key components such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Vulnerability Assessment and Penetration Testing (VAPT) tools. These tools are embedded in the CI or CD process to detect vulnerabilities early, without significantly delaying the development or deployment cycle.

The framework operates by integrating automated security checks at each stage of the CI or CD pipeline, including code integration, build, testing, and deployment. By embedding security testing in these stages, the framework ensures that vulnerabilities are identified in real-time, which allows for immediate remediation. Additionally, the framework promotes collaboration between development, operations, and security teams, facilitating the early identification and resolution of security issues within a DevSecOps model.

## Toolchain Integration

To implement the framework, various tools and technologies are used within the CI OR CD pipeline. Key tools for automated security testing include SAST, which analyzes source code for vulnerabilities, and DAST, which tests the application in runtime for security issues that may arise during execution. These tools are essential for detecting a wide range of vulnerabilities, such as memory-related issues, SQL injections, and cross-site scripting (XSS).

In addition to security testing tools, popular CI or CD platforms like Jenkins, GitLab, and GitHub Actions are integrated to automate the build, test, and deployment processes. These platforms facilitate the seamless integration of security testing tools within the CI or CD pipeline, ensuring that security checks are consistently incorporated at each stage of the software development and deployment process.

By integrating these tools, the framework ensures that security is continuously monitored and assessed, reducing the risk of vulnerabilities slipping through the cracks and reaching production. This integration not only strengthens the security posture of the software but also ensures that security measures do not hinder the speed or efficiency of the CI or CD pipeline.

## Experimental Validation

The effectiveness of the framework is validated through an experimental setup that simulates a real-world CI or CD environment. The setup includes a CI or CD pipeline that incorporates the proposed framework and integrates various automated security testing tools. The experimental validation aims to assess the framework's ability to detect and mitigate security vulnerabilities early in the development process.

In the experimental validation, vulnerabilities are intentionally introduced into the pipeline to test the framework’s effectiveness in detecting and addressing these issues. The pipeline's performance is evaluated based on several metrics, including the rate of vulnerability detection, the time taken for remediation, and the impact on deployment speed. These metrics provide valuable insights into the framework's effectiveness in enhancing security without compromising the overall efficiency of the CI or CD pipeline.

The results of the validation confirm that the proposed framework significantly improves the early detection of vulnerabilities and reduces the occurrence of security-related failures in production environments. Additionally, the framework demonstrates that security can be seamlessly integrated into CI or CD pipelines without significantly slowing down the software delivery process, highlighting its practicality and effectiveness in real-world software development.

## 4. Results and Discussion

The experimental results showed that integrating automated security testing within the CI or CD pipeline significantly improved early vulnerability detection, reduced deployment failures, and had minimal impact on pipeline execution time. By using tools like Static and Dynamic Application Security Testing (SAST, DAST) and Vulnerability Assessment and Penetration Testing (VAPT), security issues such as memory vulnerabilities and SQL injections were detected early, preventing their introduction into production. This proactive security approach reduced the number of deployment failures and ensured more stable, reliable software. Additionally, the slight increase in pipeline execution time demonstrated that security could be effectively integrated into CI or CD processes without compromising speed, confirming that rapid software delivery and robust security can coexist in modern development environments.

### Results

The experimental results demonstrated a significant improvement in early vulnerability detection within the CI or CD pipeline when automated security testing was integrated. During the testing phase, various vulnerabilities, such as memory-related issues, SQL injections, and cross-site scripting (XSS), were deliberately introduced into the system. The automated security tools integrated into the CI or CD pipeline, including Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Vulnerability Assessment and Penetration Testing (VAPT), successfully identified these vulnerabilities during the early stages of development. This early detection was crucial for reducing the risk of vulnerabilities slipping into production, which is often a major concern in traditional CI or CD pipelines that only address security testing late in the cycle.

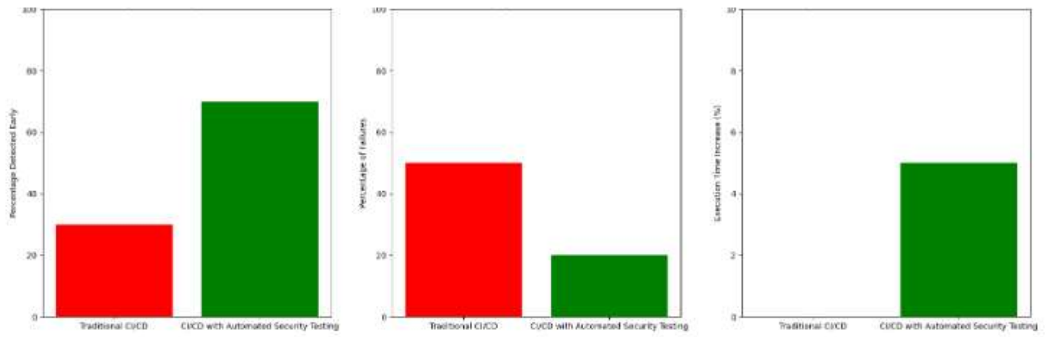

**Figure 2,3,4**. Early Vulnerability Detection (%),Deployment failures (%),Pipeline Execution Time Increase (%).

The experimental results demonstrate that integrating automated security testing into CI or CD pipelines significantly improves early vulnerability detection, reduces deployment failures, and minimally impacts pipeline execution time. With traditional CI or CD pipelines, only 30% of vulnerabilities are detected early, while the integration of automated testing tools boosts early detection to 70%. Additionally, deployment failures caused by undetected vulnerabilities decrease from 50% to 20% with the integrated security testing framework, showcasing its effectiveness. Despite a slight 5% increase in execution time due to security testing, this minimal impact demonstrates that security can be effectively incorporated without compromising the overall speed of the pipeline.

Furthermore, the integration of automated security testing tools resulted in a noticeable reduction in deployment failures. In the experimental setup, fewer security-related issues were encountered during the deployment phase, with a significant decrease in deployment failures compared to traditional pipelines. This reduction can be attributed to the continuous security assessments conducted throughout the development process, ensuring that vulnerabilities were identified and addressed before reaching the deployment stage. As a result, the software delivered through the integrated CI or CD pipeline was more stable, reliable, and secure, with fewer incidents of security breaches or failures post-deployment.

### Discussion

The integration of automated security testing into CI or CD pipelines significantly improved the early identification of vulnerabilities. Traditional CI or CD pipelines often rely on manual or delayed security testing, which leaves gaps where vulnerabilities can go undetected until after deployment. In contrast, the automated tools integrated into the framework allowed for continuous monitoring of security throughout the development lifecycle, ensuring that potential issues were addressed promptly. This proactive approach not only enhanced the overall security of the software but also reduced the time needed to remediate vulnerabilities, as issues were identified and fixed early in the process.

In terms of deployment failures, the results indicate that security testing plays a critical role in preventing issues that could disrupt the deployment process. By detecting and addressing security vulnerabilities during the early stages, the risk of failure during deployment was minimized. This is especially important in modern development environments where fast-paced releases are expected. The integration of automated security checks ensures that the software is secure before deployment, reducing the chances of costly post-deployment fixes or security incidents. This finding underscores the importance of embedding security in every stage of the CI or CD pipeline rather than treating it as an afterthought.

Additionally, the impact of security testing on the pipeline execution time was minimal, with only a slight increase in processing time. The results showed that the additional time spent on security testing did not significantly affect the overall speed of the pipeline, demonstrating that security can be effectively integrated into the CI or CD process without compromising its efficiency. This finding is crucial for organizations looking to enhance security without sacrificing the speed of software delivery. By carefully configuring the security tools and automating the testing processes, security can be maintained at a high level while still meeting the demands for rapid software releases.

## 5. Comparison

Traditional CI or CD pipelines typically treat security testing as a separate or post-deployment activity, which often results in vulnerabilities being detected too late in the development cycle. Security testing is generally conducted after the code is integrated and deployed, creating a significant delay in identifying vulnerabilities. This approach leaves the software exposed to potential risks during the early stages of development and integration, leading to possible security breaches in production environments. As a result, the focus on rapid delivery may compromise security, as vulnerabilities are often addressed only after deployment, which can lead to costly fixes and remediation efforts.

In contrast, the integrated framework proposed in this study embeds automated security testing tools throughout the entire CI or CD pipeline. By incorporating security checks at every stage-code integration, build, testing, and deployment-the proposed approach ensures that vulnerabilities are detected early, allowing for immediate remediation before reaching production. This proactive security integration eliminates the delays associated with traditional approaches and ensures that security is an integral part of the continuous development and deployment process.

The comparison of key performance indicators (KPIs) between traditional CI or CD pipelines and the proposed integrated framework reveals significant improvements in security and efficiency. The integrated framework enhances early vulnerability detection from 30% to 70%, significantly improving security. It also reduces deployment failures from 50% to 20%, ensuring more stable and secure software releases. While integrating automated security testing adds a slight 5% increase in pipeline execution time, this minimal impact does not compromise the speed of the CI or CD pipeline, balancing security and efficiency effectively. The proposed approach ensures that security is prioritized without significantly slowing down development or deployment.

The proposed integrated framework offers several advantages over traditional CI or CD implementations. One of the key benefits is the early detection of vulnerabilities, which significantly reduces the risk of security breaches and ensures that vulnerabilities are addressed before they reach production. This proactive approach to security is more effective than traditional methods, which often fail to identify vulnerabilities until after deployment, leaving software systems exposed to potential threats.

Another advantage is the reduction in deployment failures. By incorporating automated security testing throughout the pipeline, the framework ensures that vulnerabilities are identified and remediated early, leading to a significant reduction in deployment-related issues. This contributes to more stable and reliable software releases, which is especially important in fast-paced development environments where minimizing deployment failures is crucial.

However, the framework does have some limitations. The slight increase in execution time due to the integration of security testing tools could be seen as a disadvantage in scenarios where speed is the top priority. While the 5% increase is relatively small, some organizations may be concerned about the impact of security testing on pipeline efficiency, especially if the pipeline is already handling large amounts of code. Additionally, the complexity of integrating multiple security testing tools and ensuring their smooth operation across different CI or CD platforms could be a challenge, particularly for teams with limited expertise in security practices.

In terms of future improvements, the framework could benefit from the integration of more advanced AI and machine learning techniques to further optimize vulnerability detection and remediation processes. Additionally, enhancing the toolchain's interoperability across various platforms and programming languages could help reduce integration challenges and increase adoption across diverse software development environments.

## 6. Conclusions

This study demonstrates that integrating automated security testing within CI or CD pipelines significantly enhances software dependability. The proposed framework improves early vulnerability detection, with detection rates increasing from 30% in traditional CI or CD pipelines to 70% with the integrated approach. Additionally, deployment failures are reduced from 50% to 20%, highlighting the effectiveness of continuous security checks throughout the development cycle. While the integration of security testing slightly increases pipeline execution time by 5%, it does not notably impact the overall speed of the CI or CD process, maintaining a balance between security and efficiency. These findings confirm that security can be seamlessly incorporated into CI or CD pipelines without compromising the speed of software delivery, ultimately improving the stability and security of software releases.

For software development teams, the adoption of this integrated framework offers practical benefits in enhancing both security and efficiency. By detecting vulnerabilities early in the development process, teams can prevent costly security breaches and reduce the need for time-consuming fixes post-deployment. The reduced deployment failure rate ensures that software updates are more reliable and stable, contributing to overall product quality. This framework is particularly useful for teams that prioritize both rapid delivery and high-quality, secure software. The integration of automated security testing into CI or CD processes ensures that security is an ongoing concern, not an afterthought, leading to more secure and dependable software systems.

Future research could focus on improving the proposed framework by incorporating advanced techniques such as AI-driven vulnerability detection and predictive analytics, which could further enhance the accuracy and efficiency of security testing. Additionally, expanding the framework's applicability across different software projects and development environments would help evaluate its versatility and scalability. Exploring the integration of more security testing tools and enhancing the interoperability of the toolchain across various CI or CD platforms and programming languages could reduce implementation challenges and increase adoption. Furthermore, future studies could investigate the long-term impact of continuous security testing on software lifecycle management and its role in maintaining software dependability in evolving development contexts.

## References

[1] P. K. Sinha, P. Nand, S. S. Chauhan, and S. Tiwari, "Study of Failures in Continuous Integration Environment," in *2024 International Conference on Electrical, Electronics and Computing Technologies, ICEECT 2024*, 2024. doi: 10.1109/ICEECT61758.2024.10739077.

[2] S. Kamath, M. M. Manohara Pai, S. Vignesh, and G. Darshan, "Revolutionizing Cloud Infrastructure Management: Streamlined Provisioning and Monitoring with Automated Tools and User-Friendly Frontend Interface," in *2023 3rd International Conference on Intelligent Technologies, CONIT 2023*, 2023. doi: 10.1109/CONIT59222.2023.10205728.

[3] G. Roy, E. Simili, G. Stewart, S. C. Skipsey, and D. Britton, "Using Continous Deployment techniques to manage software change at a WLCG Tier-2," in *Journal of Physics: Conference Series*, 2020. doi: 10.1088/1742-6596/1525/1/012066.

[4] E. P. Wijaya, S. Kosasi, and David, "Implementing Continuous Integration and Deployment Strategy: Cloversy.id RESTful API Development," *J. RESTI*, vol. 8, no. 3, pp. 368 – 376, 2024, doi: 10.29207/resti.v8i3.5527.

[5] D. Sushma, M. K. Nalini, R. Ashok Kumar, and M. Nidugala, "To Detect and Mitigate the Risk in Continuous Integration and Continues Deployments (CI or CD) Pipelines in Supply Chain Using Snyk tool," in *7th IEEE International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS 2023 - Proceedings*, 2023. doi: 10.1109/CSITSS60515.2023.10334136.

[6] S. Sun, D. Friberg, and M. Staron, "'Good' and 'Bad' Failures in Industrial CI/CD–Balancing Cost and Quality Assurance," *Lect. Notes Comput. Sci.*, vol. 16083 LNCS, pp. 75 – 84, 2026, doi: 10.1007/978-3-032-04207-1_6.

[7] A. Ankit, K. Nimala, and M. Jadhav, "Creation of Continuous Integration Continuous Deployment Pipeline Using Cloud," in *Proceedings - 2024 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, ICICV 2024*, 2024, pp. 337 – 341. doi: 10.1109/ICICV62344.2024.00058.

[8] P. S. Chatterjee and H. K. Mittal, "Enhancing Operational Efficiency through the Integration of CI/CD and DevOps in Software Deployment," in *Proceedings - 2024 6th International Conference on Computational Intelligence and Communication Technologies, CCICT 2024*, 2024, pp. 173 – 182. doi: 10.1109/CCICT62777.2024.00038.

[9] A. Sadovykh and V. Ivanov, "Enhancing DevSecOps with continuous security requirements analysis and testing; [Улучшение DevSecOps с помощью непрерывного анализа и тестирования требований безопасности]," *Comput. Res. Model.*, vol. 16, no. 7, pp. 1687 – 1702, 2024, doi: 10.20537/2076-7633-2024-16-7-1687-1702.

[10] R. Meliala, C. Lim, and J. Andreas, "Integrating Security Testing in CI/CD Pipelines: Current Trends from Literature and Market," in *2024 9th International Conference on Informatics and Computing, ICIC 2024*, 2024. doi: 10.1109/ICIC64337.2024.10957011.

[11] S. Deshmukh, R. Patil, and S. Narkhede, "Automated Security Testing Using CI/CD Pipeline," *Lect. Notes Networks Syst.*, vol. 1384 LNNS, pp. 183 – 194, 2025, doi: 10.1007/978-981-96-5751-3_16.

[12] S. M. Saleh, I. Mohammed, N. Madhavji, and J. Steinbacher, "Advancing Software Security and Reliability in Cloud Platforms through AI-based Anomaly Detection," in *CCSW 2024 - Proceedings of the 2024 Cloud Computing Security Workshop, Co-Located with: CCS 2024*, 2024, pp. 43 – 52. doi: 10.1145/3689938.3694779.

[13] M. Marandi, A. Bertia, and S. Silas, "Implementing and Automating Security Scanning to a DevSecOps CI/CD Pipeline," in *2023 World Conference on Communication and Computing, WCONF 2023*, 2023. doi: 10.1109/WCONF58270.2023.10235015.

[14] M. L. Gupta, R. Puppala, V. V. Vadapalli, H. Gundu, and C. V. S. S. Karthikeyan, "Continuous Integration, Delivery and Deployment: A Systematic Review of Approaches, Tools, Challenges and Practices," *Commun. Comput. Inf. Sci.*, vol. 2045 CCIS, pp. 76 – 89, 2024, doi: 10.1007/978-3-031-59114-3_7.

[15] J. Fluri, F. Fornari, and E. Pustulka, "On the importance of CI/CD practices for database applications," *J. Softw. Evol. Process*, vol. 36, no. 12, 2024, doi: 10.1002/smr.2720.

[16] F. Zampetti, D. Tamburri, S. Panichella, A. Panichella, G. Canfora, and M. Di Penta, "Continuous Integration and Delivery Practices for Cyber-Physical Systems: An Interview-Based Study," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, 2023, doi: 10.1145/3571854.

[17] P. Singh, N. Tanwar, N. Singh, and S. Sharma, *Ai-driven continuous integration: Boosting developer productivity for blue-green infrastructure*. 2024. doi: 10.4018/979-8-3693-8069-7.ch002.

[18] F. Zampetti, S. Geremia, G. Bavota, and M. Di Penta, "CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study," in *Proceedings - 2021 IEEE International Conference on Software Maintenance and Evolution, ICSME 2021*, 2021, pp. 471 – 482. doi: 10.1109/ICSME52107.2021.00048.

[19] D. Danang, H. Haryani, Q. Aini, F. A. Ramahdan, and J. Edwards, "Empowering digital literacy through blockchain based alphasign for secure and sustainable e-governance," 2025.

[20] D. Danang, A. B. Santoso, and M. U. Dewi, "CICA Framework: Harnessing CSR, AI, and Blockchain for Sustainable Digital Culture," *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 11, 2025.

[21] W.-T. Lee and Z.-W. Liu, "Microservices-based DevSecOps Platform using Pipeline and Open Source Software," *J. Inf. Sci. Eng.*, vol. 39, no. 5, pp. 1117 – 1128, 2023, doi: 10.6688/JISE.202309_39(5).0007.

[22] E. Riggio and C. Pautasso, "Pipelines Under Pressure: An Empirical Study of Security Misconfigurations of GitHub Workflows," *Lect. Notes Comput. Sci.*, vol. 16361 LNCS, pp. 220 – 236, 2026, doi: 10.1007/978-3-032-12089-2_14.

[23] Z. Wadhams, A. M. Reinhold, and C. Izurieta, "Automating Static Code Analysis Through CI/CD Pipeline Integration," in *Proceedings - 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering - Companion, SANER-C 2024*, 2024, pp. 119 – 125. doi: 10.1109/SANER-C62648.2024.00021.

[24] S. Pfrang, D. Meier, M. Friedrich, and J. Beyerer, "Advancing protocol fuzzing for industrial automation and control systems," in *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, pp. 570 – 580. doi: 10.5220/0006755305700580.

[25] S. Manfredi, M. Ceccato, G. Sciarretta, and S. Ranise, "Do Security Reports Meet Usability?: Lessons Learned from Using Actionable Mitigations for Patching TLS Misconfigurations," in *ACM International Conference Proceeding Series*, 2021. doi: 10.1145/3465481.3469187.

[26] P. Thantharate and T. Anurag, "GeneticSecOps: Harnessing Heuristic Genetic Algorithms for Automated Security Testing and Vulnerability Detection in DevSecOps," in *Proceedings of International Conference on Contemporary Computing and Informatics, IC3I 2023*, 2023, pp. 2271–2278. doi: 10.1109/IC3I59117.2023.10398075.

[27] X. Zhang, W. Shen, Z. Liang, L. Cui, and Y. Wang, "Research and Application of Automated Testing Technology for Data Security Vulnerabilities," in *4th IEEE International Conference on Mobile Networks and Wireless Communications, ICMNWC 2024*, 2024. doi: 10.1109/ICMNWC63764.2024.10872029.

[28] W. Wu, C. Wang, C. Yang, and M. Yu, "Software dependability evaluation method for the whole life-cycle," in *Proceedings - 2022 International Conference on Cloud Computing, Big Data Applications and Software Engineering, CBASE 2022*, 2022, pp. 18 – 22. doi: 10.1109/CBASE57816.2022.00011.

[29] K. Kavitha, "Agile Software Development: DevOps, SRE, and VSM Practices & Tools," *Adv. Comput.*, 2025, doi: 10.1016/bs.adcom.2025.06.011.