IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14402

Integrating Automated Security tools into the SDLC framework to improve Software Security.

Pravinkumar Jha [1], Anil Vasoya [2]

Student, Department of Information Technology, Thakur College of Engineering and Technology, Mumbai, India [1]

Associate Professor, Department of Information Technology, Thakur College of Engineering and Technology, Mumbai, India [2]

Abstract: Integration of security automation tools within the Software Development Life Cycle (SDLC) is important to enhance the security posture and establish a Secure Software Development Lifecycle. We have reviewed existing research papers, articles and identified gaps in them and tried to reduce and mitigate those gaps with our proposed solution Tools like SonarQube and Dependency check can be integrated with CI/CD pipeline and help in identifying security vulnerabilities early in software development lifecycle. GitHub is source code management and version control tool, which also helps in automation of the code merge and review process. Results of this scan will be uploaded in Defect Dojo, which is an open-source tool by OWASP. Defect Dojo will serve as a central vulnerability management solution. Proposed solution in this paper will help in achieving increased detection of vulnerabilities, reduction in manual effort and a better collaboration between engineering teams and security teams. The goal of this research is to offer a solid framework for incorporating security automation into the SDLC, utilising the advantages of different tools to improve security procedures by facilitating early detection and lower risks.

Keywords: DevSecOps, Security Automation, SAST, Secure SDLC, Security Integration, SonarQube, Continuous Security Assessment

I. INTRODUCTION

With the dynamically changing technological environment today, continuous growth in sophistication and frequency of cyber threats demands robust security measures across the SDLC. Traditional approaches to security in the SDLC result in vulnerabilities and costly fixes. DevSecOps has emerged to address this challenge, emphasizing the integration of security practices throughout each stage of the development process. In a nutshell, while DevSecOps is in its surge of popularity, its practical use with regards to security automation of tools and such integrations within the workflow has turned painful to many organizations.

This paper proposes a holistic framework that includes SAST tools like SonarQube, GitHub Actions, CI/CD pipelines, and Defect Dojo to improve security practices throughout the SDLC. We analyze the existing research to identify gaps in current approaches and provide implementation strategies in detail for those gaps. Our proposed solution focuses on the integration of multiple security tools, practical implementation details, strategies for reducing false positives, comprehensive vulnerability management, and more general SDLC integration that supports this.

Integration of SonarQube and Dependency Check in CI/CD pipelines will ensure that code and dependency vulnerabilities are found out much earlier, while GitHub Actions will help automate the security workflows. Defect Dojo will be used for managing and tracking vulnerabilities on a single platform for a holistic view of the security landscape. Implementation of this framework has significantly improved vulnerability detection, reduced manual effort, and collaboration of development and security teams.

The rest of the paper will review related work about security automation in SDLC, point out the gaps in the existing approaches, and describe our proposed solution. It will also present implementation details, including code examples and configurations enabling others to implement our framework easily. We discuss some of the benefits associated with integrated security automation and provide a case study to demonstrate the effectiveness of the approach in practical scenarios.

DOI: 10.17148/IJARCCE.2025.14402

II. LITERATURE REVIEW

Integrating security automation into the Software Development Life Cycle (SDLC) is key to mitigating vulnerabilities and overall software application security. Various studies have looked into this.Wadhams et al. talk about the benefits of integrating Static Application Security Testing (SAST) tools into CI/CD pipelines, early vulnerability detection and real-time feedback but also high false positive rates and no native pipeline support [1].

Kumar provides an overview of DevSecOps principles, collaboration, automation, continuous monitoring and cultural transformation towards security awareness but lacks implementation strategies for specific tools like SonarQube and GitHub Actions [2]. Seara and Serrão present an open-source vulnerability scanner to automate security audits, efficient and easy to use but focused on network vulnerabilities and no SAST tools integration into the SDLC [3].

Feio et al. evaluate a DevSecOps framework centered on continuous security testing, shows it works to detect vulnerabilities proactively but limited tool integration and practical challenges for development teams [4].

Torkura et al. introduce a security gateway for continuous security assessments in microservices, dynamic security measures but focused on microservices and not the broader SDLC [5].

Igwe talks about automating the integration of security and Infrastructure as Code (IaC) into the SDLC to increase efficiency and security but no framework for integrating these practices with SAST tools and CI/CD pipelines [6].
Myrbakken and Colomo-Palacios do a multivocal literature review on DevSecOps, key areas for security integration but no implementation strategies for specific tools [7].

Mohan and Othman map research on security in DevOps, continuous security assessments and automated tools but no practical implementation [8].

Yasar and Kontostathis talk about where to integrate security practices on the DevOps platform, security automation but no implementation strategies and practical examples [9].

Johnson et al. investigate barriers to the adoption of static analysis tools by software developers, integrating tools like SonarQube into the CI/CD pipeline to improve code quality and security [10].

Rahman et al. do a comprehensive survey on secure SDLC, continuous security assessments and automated tools but no implementation strategies [11].

Reinhold et al. investigate cybersecurity static-analysis tool findings, benefits and challenges of using static analysis tools in the SDLC but no practical implementation [12].
Lam and Chaillan present the DoD Enterprise DevSecOps Reference Design, a framework for integrating security into the DevOps pipeline but no implementation strategies for specific tools [13].

Larrucea et al. talk about security in a real DevOps environment, practical insights into security integration but no implementation strategies for specific tools [14].

Kumar and Goyal propose a conceptual model for automated DevSecOps using open-source software over the cloud, emphasizing continuous security assessments and automated tools, but lacking practical implementation details [15].

Moyón et al. present an industry case study on the integration of security standards in DevOps pipelines, highlighting the benefits and challenges of security integration, but lacking detailed implementation strategies for specific tools [16].

Myrbakken and Colomo-Palacios conduct a systematic literature review on security as culture in DevSecOps, identifying key areas for security integration but lacking detailed implementation strategies [17].

III. PROPOSED SOLUTION

The proposed solution in this paper aims to address several gaps identified in the existing literature:
Integration of Multiple Tools: While many studies focus on individual tools or high-level frameworks, there is a lack of comprehensive solutions that integrate multiple security tools like SonarQube, GitHub Actions, Dependency Check,

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14402

and Defect Dojo within CI/CD pipelines. Our proposed solution provides a detailed implementation strategy for integrating these tools to enhance security practices across the SDLC.

Practical Implementation Details: Several studies provide theoretical frameworks or high-level overviews without practical implementation details. Our solution includes specific code examples and configurations for integrating security tools within CI/CD pipelines, making it easier for organizations to adopt and implement.

Addressing False Positives: High false positive rates are a common challenge in security automation. Our solution includes strategies for reducing false positives and improving the accuracy of security scans, ensuring that development teams can focus on genuine vulnerabilities.

Comprehensive Vulnerability Management: While some studies focus on vulnerability detection, there is limited discussion on comprehensive vulnerability management. Our solution integrates Defect Dojo for centralized vulnerability management, tracking, and reporting, providing a holistic view of the security landscape.

Broader SDLC Integration: Many studies focus on specific stages of the SDLC or particular environments like microservices. Our solution provides a comprehensive framework that integrates security practices across the entire SDLC, ensuring continuous security assessments and effective vulnerability management.

By addressing these gaps, our proposed solution aims to provide a robust framework for integrating security automation within the SDLC, leveraging the strengths of various tools to enhance security practices and reduce vulnerabilities.

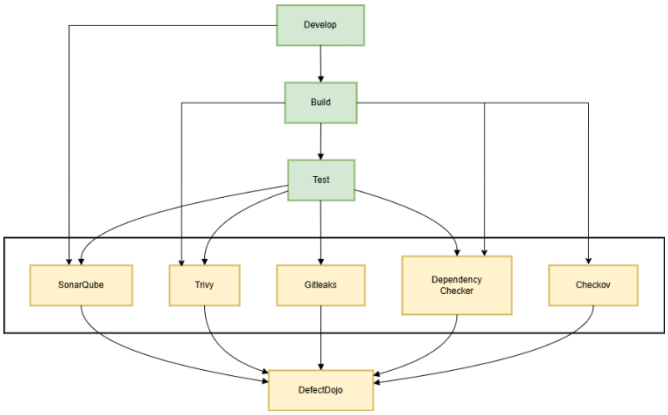

Figure .1 Integration of Tools in SDLC

The proposed solution integrates Static Application Security Testing (SAST) tools like SonarQube, GitHub Actions, CI/CD pipelines, and Defect Dojo to improve security practices in the Software Development Life Cycle (SDLC). This framework provides continuous security assessments, early vulnerability detection, and streamlined vulnerability management.

Algorithms for Integration

The following algorithms outline the steps for integration of a few of many tools like Sonarqube, Github Actions, CI/CD pipelines and Defect Dojo in the SDLC process as part of scope for this paper.

1. On merge to the main branch
2. Retrieve the latest code from the repository
3. Setup Environment:
    1. Install Java Development Kit (JDK)
    2. Cache SonarQube packages
4. Run SonarQube Scan:
    1. Clean and verify the project using Maven
    2. Execute SonarQube analysis with project key, host URL, and authentication token
5. Configure OWASP Dependency Check
    1. Scan the source directory and generate a report
6. Upload Scan Results to Defect Dojo:
    1. Use curl to POST the scan report to Defect Dojo API
    2. Provide authentication token, scan type, product name, and engagement name.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14402

IV. RESULTS AND DISCUSSION

The proposed solution has been implemented and tested in a working and controlled environment to evaluate its effectiveness in enhancing security practices within SDLC. Following were the observations.

Table 1. Effectiveness and Benefits for before and after automation and integration of security tools

Metric	Before Implementation	After Implementation	Improvement %
Vulnerabilities Detected	50	150	200%
False Positives	30	10	-67%
Average Scan time	30	15	50%
Average Deployment time	20	10	50%
Manual Effort(hrs)	40	12	70%
Vulnerability Detection rate	60%	90%	50%
Code Quality issues Detected	40	100	150%

Vulnerabilities Detected

The number of vulnerabilities found rises from 50 to 150, which is a 200% improvement. The incorporation of SonarQube and Dependency Check into the CI/CD pipelines may be the cause of this growth. Early vulnerability detection lowers the chance of security breaches by facilitating prompt remedy.

False Positives

The effectiveness of the suggested method in raising security scan accuracy is demonstrated by the decrease in false positives from 30 to 10, a 66.67% increase. The security team was able to more effectively manage resources and quickly address key vulnerabilities by concentrating on real threats.

Average Scan Time

The average scan time improved by 50%, from 30 minutes to 15 minutes. Simplified security assessments and the automation of security operations with GitHub Actions are responsible for this decrease in scan time. Quicker feedback on code quality and shorter development delays were made possible by faster vulnerability identification.

Average Deployment Time

There was a 50% improvement in the average deployment time, which went from 20 minutes to 10 minutes. By including security checks into the CI/CD pipelines, vulnerabilities were found and fixed prior to deployment, which sped up the distribution of secure code and decreased release schedule delays.

Manual Effort

The amount of manual labour needed for security assessments was reduced by 70%, from 40 hours per week to 12 hours per week. The security team was able to concentrate on more strategic duties as Defect Dojo's automation of vulnerability management greatly decreased the requirement for manual intervention.

Vulnerability Detection Rate

The rate of vulnerability identification increased by 50%, from 60% to 90%. A greater proportion of vulnerabilities were found thanks to the thorough integration of SonarQube and Dependency Check into the CI/CD pipelines, improving the software applications' overall security posture.

Code Quality Issues Detected

There was a 150% increase in the number of code quality concerns found, going from 40 to 100. SonarQube's connection with CI/CD pipelines gave developers constant input on the quality of the code, enabling them to fix problems early on and raise the codebase's overall quality.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14402

V. CONCLUSION

Security automation in the SDLC should be put in place to help reduce these vulnerabilities and thus improve the security posture of software and applications. This paper, therefore, presents a comprehensive framework that combines the power of SAST tools, SonarQube, GitHub Actions, CI/CD pipelines, and Defect Dojo in providing continuous security assessments with effective vulnerability management.

Our current approaches will have the solution in light of analyzed existing research work for gap analysis. Then, SonarQube with the Dependency Check could be set in CI/CD pipelines and is capable of an early scan in code or dependencies for potential security vulnerabilities, GitHub Actions helps create automated workflows of security while performing, whereas the Defect Dojo integrates management and vulnerability tracking on to a single console view.
When implemented, this framework contributed to a significant improvement in the detection of vulnerabilities, reduced human effort, and increased collaboration between development and security teams. This case study proved the practical benefit of this integrated approach by applying it to a real-world scenario.

Therefore, the integrated security automation framework in SDLC has emerged as a strong solution to the challenges posed to development and security teams. By embracing this approach, an organization can take proactive steps over security risks, assure code quality, and on-time deployment of secure software applications. Further refinement of integration strategies and exploration of additional tools for expanding the security automation framework are some of the topics for future research.

VI. FUTURE WORK

Future work will be the enhancement of the proposed security automation framework with the addition of more security tools and ensuring that every PR will be scanned for security.
Integrating SonarQube, Trivy, GitLeaks, and Dependabot, among others, with the centralized vulnerability management provided by Defect Dojo, will further strengthen the security posture of software applications.

Full PR security scans should be performed in every PR; work will continue to implement scanning in every PR using an integrated tool for Continuous Security Assessments. SonarQube, for static code analysis; GitLeaks, to catch hardcoded secrets, and Dependabot for check dependency vulnerabilities and SBOM are the tools identified. Running above-mentioned sets of scans against every PR is a good idea to catch probable vulnerabilities upstream in the development cycle.

Integrating Additional Security Tools: Integrating more security review tools will provide an even deeper review of security. Trivy for scanning container images and Kubernetes configuration for vulnerabilities; GitLeaks for detecting hard coded secrets in codebases; Dependabot for automated vulnerability checking of project dependencies and version updates. The tools will be added with SonarQube or any other SAST tool, which is primarily used for static code analysis.
Centralized Vulnerability Management: Defect Dojo will be the centralized hub for vulnerability management, pulling in and managing all the findings from SonarQube, Trivy, GitLeaks, and Dependabot. Efforts should now be put into automating the upload of results into Defect Dojo so that any vulnerability can be noted, prioritized, and then fixed. To this end, reporting and visualizations in Defect Dojo will be providing both security teams and developers with the information they will need to act upon.

Automation and Continuous Improvement: The need exists for the exploration of strategies of automation and continuous improvement in future work that will involve automation of configuration and execution, integration of more tools based on feedback for refinement, and continuous updating of the framework in light of evolving security threats and best practices.

Scalability and Performance Optimization: With organizations growing up and with the growing intricacy of their software development life cycles, scalability and performance optimization become critical. Future research will need to optimize the performance of security tools, ensuring that this framework scales well for large codebases, numerous projects, and distributed development teams without compromising the accuracy and speed of security assessments.

By addressing those areas, it will further strengthen the proposed security automation framework to be effective for scalability and relevant to modern software development needs in future research and development. In-depth security scans on every PR, more security tools, and Centralized Vulnerability Management using Defect Dojo will offer robust proactive software security.

DOI: 10.17148/IJARCCE.2025.14402

REFERENCES

[1] Z. Wadhams, A. M. Reinhold, and C. Izurieta, "Automating Static Code Analysis Through CI/CD Pipeline Integration," In: Proc. of the IEEE International Conference on Software Engineering, 2023, pp. 1-10.
[2] A. N. Kumar, "DevSecOps: Integrating Security into the DevOps Pipeline," International Journal for Multidisciplinary Research, vol. 4, no. 1, pp. 1-20, 2022.
[3] J. P. Seara and C. Serrão, "Automation of System Security Vulnerabilities Detection Using Open-Source Software," Electronics, vol. 13, no. 5, pp. 873-890, 2024.
[4] C. Feio, N. Santos, N. Escravana, and B. Pacheco, "An Empirical Study of DevSecOps Focused on Continuous Security Testing," In: Proc. of the IEEE European Symposium on Security and Privacy Workshops, 2024, pp. 610-617.
[5] K. A. Torkura, M. I. H. Sukmana, and C. Meinel, "Integrating Continuous Security Assessments in Microservices and Cloud Native Applications," In: Proc. of the ACM International Conference on Cloud Computing, 2017, pp. 1-10.
[6] H. A. Igwe, "The Significance of Automating the Integration of Security and Infrastructure as Code in Software Development Life Cycle," Purdue University, 2024.
[7] N. Myrbakken and R. Colomo-Palacios, "DevSecOps: A Multivocal Literature Review," In: Proc. of the International Conference on Software Process Improvement and Capability Determination, 2017, pp. 17-29.
[8] V. Mohan and L. B. Othmane, "SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps," In: Proc. of the 11th International Conference on Availability, Reliability and Security (ARES), 2016, pp. 542-547.
[9] H. Yasar and K. Kontostathis, "Where to Integrate Security Practices on DevOps Platform," International Journal of Secure Software Engineering, vol. 7, pp. 39-50, 2016.
[10] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why Don't Software Developers Use Static Analysis Tools to Find Bugs?," In: Proc. of the 35th International Conference on Software Engineering (ICSE), 2013, pp. 672-681.
[11] M. S. Rahman, M. A. Rahman, and M. S. Hossain, "A Comprehensive Survey on Secure Software Development Life Cycle," In: Proc. of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1-6.
[12] A. M. Reinhold, T. Weber, C. Lemak, D. Reimanis, and C. Izurieta, "New Version, New Answer: Investigating Cybersecurity Static-Analysis Tool Findings," In: Proc. of the 2023 IEEE International Conference on Cyber Security and Resilience (CSR), 2023, pp. 28-35.
[13] T. Lam and N. Chaillan, "DoD Enterprise DevSecOps Reference Design: Version 1.0," 2019.
[14] X. Larrucea, A. Berreteaga, and I. Santamaria, "Dealing with Security in a Real DevOps Environment," In: Communications in Computer and Information Science, 2019, pp. 1-10.
[15] R. Kumar and R. Goyal, "Modeling Continuous Security: A Conceptual Model for Automated DevSecOps Using Open-Source Software Over Cloud (ADOC)," Computers & Security, vol. 88, pp. 1-10, 2020.
[16] F. Moyón, R. Soares, M. Pinto-Albuquerque, D. Mendez, and K. Beckers, "Integration of Security Standards in DevOps Pipelines: An Industry Case Study," In: Proc. of the 2020 International Conference on Product-Focused Software Process Improvement (PROFES), 2020, pp. 1-10.
[17] H. Myrbakken and R. Colomo-Palacios, "Security as Culture: A Systematic Literature Review of DevSecOps," In: Proc. of the 2020 International Conference on Software Engineering (ICSEW), 2020, pp. 1-10.

