IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429 Volume 5, Issue 13, April 2025 Impact Factor: 7.67

AI-Powered Devsecops: Embedding Security in CI/CD with Automation and Insight

Mrs. A. Jancy[1], R. Lokeshwaran[2], T. Javith Naseem[3]

Assistant Professor, Department of Information Technology[1]

Students, B.Tech., Final Year, Department of Information Technology[2,3]

Anjalai Ammal Mahalingam Engineering College, Thiruvarur, India

Abstract: *This paper explores the practical implementation of a DevSecOps pipeline enhanced with artificial intelligence (AI) to foster robust security, compliance, and automation across the entire software development lifecycle (SDLC). By merging development, security, and operations into an integrated framework, DevSecOps ensures proactive threat mitigation without hindering development speed. The core objective is to embed security checks directly into development and deployment workflows, leveraging automation tools and AI capabilities for early risk detection and remediation. Key tools such as GitHub Actions, SonarQube, Trivy, OWASP ZAP, and Semantic Kernel, along with Kubernetes and Docker, are utilized for secure CI/CD practices. This AI-empowered pipeline supports advanced features like vulnerability summarization, automated compliance monitoring, and scalable microservice deployments, providing a comprehensive and adaptive solution for modern software teams.*

Keywords: DevSecOps, Secure SDLC, GitHub Actions, OWASP ZAP, Trivy, Docker, Security Automation, AI Security Reporting, CI/CD, Kubernetes

I. INTRODUCTION

DevSecOps, short for Development, Security, and Operations, extends the traditional DevOps framework by embedding security practices throughout the entire software development lifecycle. In contrast to legacy security approaches that introduced checks at the end of the development phase, DevSecOps integrates security considerations from the start.

Security threats today are not only more frequent but also more sophisticated. Delayed detection often results in costly fixes and compliance issues. DevSecOps seeks to resolve these problems by making security an integral part of the development culture. Automated security scans, real-time threat analysis, and AI-assisted vulnerability assessments are key enablers of this transition.

II. DEVSECOPS TOOLS AND WORKFLOW

The DevSecOps lifecycle involves a combination of tools and automation to maintain security integrity. Below are key components integrated within the system:

Static Application Security Testing (SAST):

SonarQube is used to statically analyze the source code, helping identify security flaws, code smells, and bugs before they reach production. These scans are triggered during the development stage, ensuring real-time feedback within the CI pipeline.

Dynamic Application Security Testing (DAST):

OWASP ZAP performs simulated attacks on the running application in staging environments. This real-time penetration testing uncovers vulnerabilities that cannot be detected through static analysis alone.

Container Security:

Trivy and Snyk are employed to scan Docker images and open-source dependencies for known vulnerabilities and misconfigurations. These tools integrate into the CI pipeline to assess the security posture of containers before deployment.

CI/CD Automation:
GitHub Actions and Azure DevOps manage the build, test, and deployment workflows. Stages include static scans, containerization, SBOM creation, deployment to Kubernetes, and notification dispatch.
Infrastructure as Code (IaC) Scanning:
Checkov scans Terraform and Kubernetes manifests to ensure infrastructure adheres to secure configurations. This ensures cloud environments are provisioned securely from the beginning.
Compliance & Governance:
The pipeline includes compliance verification using OpenSCAP and InSpec, which validate infrastructure and application configurations against industry standards such as ISO 27001, GDPR, and NIST

III. MICROSERVICES SECURITY MODEL

The system is designed using a modular microservices architecture. Each service—such as user, product, and order—is deployed in isolated containers, ensuring service-level security and fault tolerance. Key security measures include:

- Securing API Gateway and Routing Logic.
- Dockerfile and Kubernetes YAML Scanning.
- Static analysis of Dockerfiles and Kubernetes deployment manifests.
- Secrets Management & Token Validation.

Each microservice not only undergoes unit testing but also produces AI-generated vulnerability summaries. These summaries are automatically posted as GitHub issues, encouraging proactive remediation.

Benefits

- Enhanced service isolation and access control
- Scalable and independent security workflows
- Real-time vulnerability insights via Semantic Kernel–GPT integration

IV. EXISTING SYSTEM

Legacy software development processes typically integrate security as a final step, resulting in delayed discovery of vulnerabilities, higher fix costs, and slower delivery timelines. These systems lack security integration within CI/CD pipelines, relying heavily on manual and reactive testing. Such environments struggle to handle the security demands of containerized applications and IaC. Additionally, fragmented logging and monitoring prevent effective incident response and real-time threat detection.

Demerits:

- Late-stage vulnerability detection
- Manual testing as a bottleneck
- Inconsistent compliance enforcement
- Poor visibility into infrastructure and containers
- Inadequate scalability for agile teams

V. PROPOSED SYSTEM

The proposed system adopts a DevSecOps approach that integrates security practices into every phase of the development pipeline. It includes continuous integration and continuous deployment (CI/CD) with automated security testing to detect issues early. Key enhancements include container image scanning with Trivy, static code analysis using SonarQube, infrastructure compliance checks, and real-time monitoring. Role-based access control (RBAC) and secure authentication using OAuth2 ensure proper user access. Security tools like OWASP ZAP and Snyk are embedded into the pipeline, ensuring dynamic and dependency scanning. Kubernetes orchestrates deployment, and monitoring tools like Azure Monitor and Wazuh enable incident detection and response, resulting in a secure, scalable, and compliant environment.

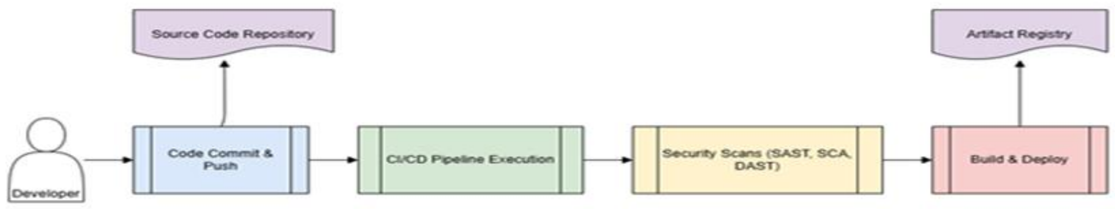
Key Features of Proposed System:

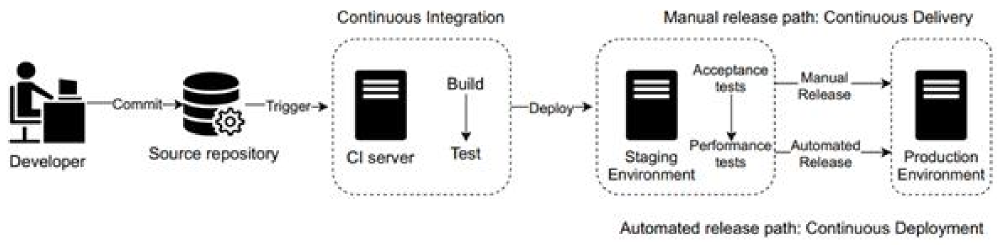
- GitHub-integrated CI/CD workflows.
- Automated SBOM generation and visualization using Dependency-Track.
- GPT-powered vulnerability summaries and remediation suggestions.
- AI-assisted dashboards for real-time insights and reporting.
- Support for hybrid and multi-cloud environments.

Advantage of Proposed System:

- Immediate feedback loops
- Automated risk scoring
- Enhanced compliance tracking
- AI-assisted remediation
- Support for both cloud-native and on-premise environments

VI. SYSTEM ARCHITECTURE DIAGRAM





VII. MODULES

List of Modules:

• Authentication and Access Control
• CI/CD Pipeline Security
• Security Scanning Module
• Compliance Monitoring
• Incident Response & Logging

1) Authentication and Access Control:
The system incorporates secure access mechanisms using OAuth2 and Role-Based Access Control (RBAC). Support for enterprise-grade identity management—via LDAP, SSO, and Active Directory—is included. Sensitive features such as policy editing and scan control are accessible only to authorized users.

2) CI/CD Pipeline Security:
The pipeline incorporates security at each CI/CD stage:
• Code retrieval and build validation.
• Static and dependency checks (SonarQube, Snyk).
• Container scanning with Trivy.
• Conditional deployment into Kubernetes based on pass/fail thresholds.

3) Security Scanning Module:
The system integrates a multi-dimensional security scanning architecture:
• Static Application Security Testing (SAST): SonarQube scans the codebase for insecure patterns
.• Software Composition Analysis (SCA): Snyk audits third-party dependencies for known vulnerabilities.
• Dynamic Application Security Testing (DAST): OWASP ZAP simulates attacks on live environments to uncover runtime flaws.
These scans run continuously—both in development and as part of automated CI/CD processes.

4) Compliance Monitoring:
Tools like OpenSCAP and InSpec automatically generate compliance reports. These validate the environment against multiple regulatory standards, making the system enterprise-ready.

5) Incident Response & Logging:
Using Azure Monitor and the ELK stack (Elasticsearch, Logstash, Kibana), the project achieves:
• Real-time threat alerting
• Centralized log aggregation
• Visual dashboards for anomaly detection Event correlation helps detect breaches or misconfigurations promptly, triggering automated incident response scripts if thresholds are breached.

VIII. CONCLUSION

This project illustrates a fully operational AI-enhanced DevSecOps pipeline that integrates security seamlessly into the software lifecycle. By using automation tools and AI models, we detected and addressed security risks proactively. Key tools included Docker, GitHub Actions, Trivy, SonarQube, and Dependency-Track, supported by GPT-based summaries.
The use of Kubernetes for orchestrating secure, scalable deployments, and real-time observability tools for monitoring, ensures that security does not slow development. The result is a resilient, compliant, and high-quality software pipeline aligned with modern security demands..

IX. FUTURE WORK

Future improvements may include expanding DevSecOps practices to multi-cloud environments (AWS, Azure, GCP) and integrating predictive analytics using AI/ML models for proactive threat detection. Enhancements like Kube-hunter, Falco, and zero-trust principles can elevate runtime security.
Full-stack SBOM coverage for languages like Go, Ruby, and Rust, along with broader compliance support (e.g., CCPA, CMMC), will further mature the pipeline. AI-driven dashboards for real-time insight and automated security decisions could help achieve autonomous DevSecOps workflows.
.

REFERENCES

[1].U.S. Department of Defense. DevSecOps Reference Design v2.0.Defense Innovation Board., 2020

[2]. Zaydi, M., Ahmed, K., & Qureshi, M. DevSecOps Practices forITSM: Integrating Security with Service Workflows. Journal of IT Security& Compliance, 8(2), 2021

[3].Smith, J., Patel, R., & Garcia, L.Security Automation inDevSecOps CI/CDPipelines: A Tool-Based Analysis. International Journalof Software Security, 12(1)., 2022

[4].Singh, A., Kumar, P., & Zhao, Y. Automated VulnerabilityManagement in DevSecOps Pipelines Using AI-Based Prioritization.Journal of Cybersecurity Engineering, 14(3), 120–135 ,2022.

[5].Patel, S., Arora, D., & Fernandes, L.Real-Time AnomalyDetection in DevSecOps Pipelines Using AI-Driven Observability Platforms. Cybersecurity Intelligence Review, 10(4), 78–91",2022.

[6].Veeramachaneni, V. A Systematic Review of DevSecOps: BridgingSecurity and Agile Development for Resilient Software Systems.International Journal of Computer Applications in Technology,2023.

[7].Rajapakse, H., Lin, W., & Rao, S.. Challenges in DevSecOpsAdoption: An Empirical Study on Tooling and Organizational Change.Cybersecurity Journal, 9(4).,2023

[8]. Martínez, L., Shah, M., & Gupta, RSBOM-Driven DevSecOps:Improving Transparency and Compliance Through Automated SoftwareBill of Materials Generation. Proceedings of the DevSecOps EuropeConference,2023.

[9].Feio, C., Santos, N., Escravana, N., & Pacheco, B. An EmpiricalStudy of DevSecOps Focused on Continuous Security Testing.Proceedings of the 2024 IEEE European Symposium onSecurity andPrivacy Workshops (EuroS&PW) 2024.

[10]. Fu, M., Pasuksmit, J., &Tantithamthavorn, C. AI for DevSecOps:A Landscape andFuture Opportunities 2024.