JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

ADAPTIVE SECURITY TESTING SCHEDULER FOR DEVSECOPS PIPELINES

[1]Shyam Srikant Bharadwaj, [2]T Harika, [3]U Jayanth, [4]Dr Rakshitha Kiran P

[1]Student, [2]Student, [3]Student, [4]Assistant Professor

Department of Computer Science & Engineering (Cyber Security),

Dayananda Sagar College of Engineering, Bengaluru, India

Abstract: The current DevSecOps pipelines frequently suffer from inefficiencies because of blanket security scans on every commit, resulting in high time complexity, unwanted scans and tests, and increased false positive cases. This paper aims to solve these by introducing a mathematical function based adaptive scheduler that efficiently and smartly selects only needed and necessary scans using scoring, thresholding, optimizing and probability based models. Therefore, This approach integrates SAST, DAST, IaC, dependence, and container scan tools, reducing additional costs and resources while providing quality. Security analysis also provides resistance against evasion and false positives, with a run time approximately reduced to 40%. Additionally, It delivers scalable, auditable, and efficient DevSecOps pipeline which can be used in all enterprises and regulated sectors. This mark the upcoming generation of safe CI/CD.

Index Terms - Adaptive Security Scheduling, Mathematical Decision Engine, Probabilistic Scan Selection, Bayesian Risk Modeling, Optimization-Based Pipeline Security, Policy-as-Code Framework, Forensic-Aware DevSecOps, AI-Augmented Security Testing, Context-Aware Vulnerability Detection, Compliance- Driven Adaptive Testing

I. INTRODUCTION

Modern software development focuses heavily on speed and flexibility, especially with the use of CI/CD pipelines. However, bringing strong security measures into these workflows without slowing development remains a major challenge. This is where DevSecOps plays an important role. Instead of adding security as an afterthought, DevSecOps aims to integrate it into every step of the development process.

In many traditional setups, pipelines still rely on broad, static scanning methods. Every commit, no matter how small or large, goes through the entire set of security tests. This greatly increases runtime, consumes more system resources, and frustrates developers because of repeated false alarms and redundant scans.

Adaptive security testing solves these issues by adjusting the depth and type of scans depending on what has changed in each commit. High-risk or major updates get full, detailed analysis, while smaller, low-impact changes receive faster, lighter scans. Studies show that DAST tools can take hours to run, even though the builds themselves complete in minutes—creating a clear bottleneck in the process. Adaptive scheduling reduces repeated scans, makes better use of resources, and encourages developers to follow secure coding practices.

The proposed research introduces a rule-based adaptive scheduler for DevSecOps pipelines that integrates multiple security tools—SAST, DAST, IaC analysers, container scans, and dependency checks—into one unified CI/CD workflow. The scheduler uses predefined rules to determine which scans are needed for each commit. This helps maintain strong security coverage while reducing overall runtime. The approach advances DevSecOps maturity by solving two main problems at once: improving pipeline performance and ensuring strong security assurance. It also sets a foundation for future upgrades using AI-driven adaptive models.

As software systems continue to move toward cloud-native and microservice architectures, the need for intelligent, context-aware security automation becomes even more critical. The adaptive scheduler addresses this by applying mathematical and probabilistic models that analyze code changes, dependencies, and historical vulnerability data. Using this information, the system can focus resources on higher-risk areas, improve efficiency, and support compliance with frameworks like OWASP and NIST.

Ultimately, this research seeks to balance fast development with continuous, automated security. The proposed adaptive scheduler aims to make security testing both smarter and more efficient, ensuring DevSecOps pipelines remain scalable, reliable, and secure across different environments.

II. LITERATURE SURVEY

Research on DevSecOps pipelines shows both the strengths of current methods and the weaknesses that still exist. Many studies highlight that using uniform, non-adaptive scanning workflows can slow down delivery and create extra friction for developers. For example, Rajapakse et al. (2021)[14] presented a systematic review showing that running the same tests for every commit, even for small changes, adds major delays to CI/CD processes.

Earlier DevSecOps research mainly focused on integrating tools, such as combining static (SAST) and dynamic (DAST) analysis within automated pipelines. Although these integrations improved overall security coverage, they did not reduce redundant scans. Aljohani and Alqahtani (2023)[15] pointed out that DAST executions take a large amount of time and stressed the need for adaptive scheduling. Malik (2023)[3] proposed a scalable DevSecOps framework, showing that pipeline scalability depends on how granular and efficient the security automation is.

Recent studies move toward adaptive and context-aware testing. Feio et al. (2024)[2] carried out an empirical study highlighting the benefits of continuous security testing and selective scan execution. Paidy (2023)[5] showed that AI-assisted adaptive testing can reduce false positives and ease developer fatigue. Similarly, Yelkoti (2025)[4] promoted the "Security as Code" approach, which builds adaptive risk management directly into pipeline definitions.

Beyond adaptivity, some works discuss the challenges of tool interoperability and the problem of too many alerts. Ramos and Yoo (2023)[16] performed a systematic review showing that false positives continue to reduce trust in automated security tools. Bernardo (2022)[8] suggested using quality gates and automated rollback strategies to manage these errors better. Papalia (2025)[7] further extended this by introducing forensic-aware mechanisms that improve traceability during incident response.

Overall, the literature shows that adaptive and intelligent pipeline security approaches are still early in development but urgently needed. While many researchers recognize inefficiencies in current methods, only a few provide concrete adaptive scheduling models. Together, these studies form the foundation for this research, highlighting the gaps in efficiency, scalability, and reliability in modern DevSecOps pipelines. Table~1 summarizes the key findings from the reviewed studies.

Table 1. Summary of literature on DevSecOps and adaptive security testing

Study	Focus area	Key findings	Limitations
Putra & Kabetta (2022)	SAST + DAST integration	Improves coverage	Increases Runtime
Rajapakse et al. (2021)	Systematic review	Identifies redundancy issue	No Adaptive Model
Feio et al. (2024)	Empirical study &	Advocates continuous adaptive testing	Limited tool support
Paidy (2023)	AI-driven adaptive testing	Reduces false positives	Needs ML Training
Bernardo (2022)	False positive management	Introduces rollback strategies	Narrow applicability

III. MOTIVATION FOR ADDING MATHEMATICAL FUNCTIONS

Adaptive security schedulers today often depend on simple rules or tool-specific settings. While these methods can work to some extent, they lack precision when dealing with different commit complexities or varying pipeline loads. Mathematical functions offer a more formal and reliable way to make such decisions, helping the scheduler find a good balance between speed, accuracy, and coverage.

For example, threshold functions can decide whether a light or full scan is needed based on the ratio of modified security-critical files to the total number of changed files in a commit. Weighted scoring models can also bring together different factors—like commit metadata, past vulnerability frequency, and the time cost of scans—into one overall decision score.

Mathematical optimization functions are also helpful when distributing limited computing resources. When several scans compete for the same resources, scheduling can be treated as a minimization problem, where the goal is to reduce total pipeline runtime while still keeping enough security coverage. Probabilistic methods such as Bayesian updating help the scheduler keep learning from real-time outcomes of earlier pipeline runs, continuously improving decision accuracy.

Without these mathematical foundations, adaptive schedulers risk becoming inconsistent or overly based on intuition. They might also favor short-term performance at the cost of reliability. Mathematical functions bring scalability and consistency across different environments and reduce the dependency on manually created rules. As more pipelines begin to include AI-based detection models, these mathematical methods will act as the bridge between human logic and machine-learned predictions, allowing smarter and more dynamic scheduling.

In short, the goal is to move adaptation from an intuitive, rule-based process to one built on clear, data-driven logic. By applying ideas from optimization, probability, and scoring theory, we aim to build DevSecOps pipelines that are efficient, secure, and dependable.

IV. PROPOSED MATHEMATICAL FUNCTIONS

4.1 Decision Score

The Decision Score (DS) function evaluates commits using weighted parameters:

$$DS = \sum_{i=1}^{n} w_i \cdot p_i \quad (1)$$

Where:

- DS= decision score
- w_i= weight of factor i
- p_i= probability of factor i
- n= total number of factors

Here, w_i represents the weight given to each parameter p_i (for example, code criticality, amount of code modified, or the history of vulnerabilities in that module). Commits with higher DS values trigger complete security scans, while those with lower scores only run lighter checks.

4.2 Thresholding Mechanism

Threshold functions classify commits into different risk categories.

- If DS < α: run only SAST.
- If $\alpha \leq$DS < β: run SAST + dependency scanning.
- If DS $\geq \beta$: run the full set of scans including DAST and container scanning.

This layered approach avoids unnecessary DAST runs, which are known to slow down pipelines.

4.3 Resource Allocation

Resource allocation is defined as:

$$\min \quad \sum_{j=1}^{m} C_j \, T_j \quad (2)$$

$$\text{Subject to: } \sum \text{Coverage}_j \geq \gamma \quad (3)$$

The model calculates the total security cost using the following variables:

- C_j : cost of test j
- T_j : time required for test j
- m : total number of security tests

In this formula, C_j refers to the cost of scan j, T_j is the runtime, and γ represents the minimum coverage required. This design helps achieve a balance between performance and security by using available resources efficiently.

4.4 Probabilistic Updating

Bayesian inference is used to continuously update the probability of vulnerabilities in a commit:

$$P(V|D) = \frac{P(D|V) \cdot P(V)}{P(D)} \quad (4)$$

Where:

- V = existence of a vulnerability
- D = observed data from the commit
- P(V|D) = updated (posterior) probability of a vulnerability given the commit data
- P(D|V) = likelihood of observing the data if a vulnerability exists
- P(V) = prior probability of a vulnerability
- P(D) = marginal probability of the observed data

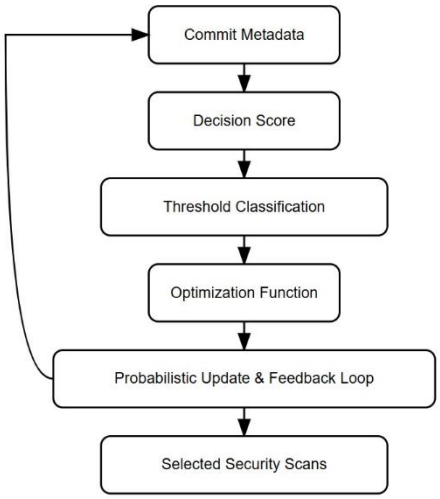

Figure 1

4.5 Integration into Pipeline

These functions are implemented within a YAML-based policy engine. Each time a new commit is made, the scheduler performs the following steps:

- Calculates DS using Eq.(1)
- Applies the thresholding rules.
- Solves the optimization problem defined in Eq.(2)
- Updates the probabilistic model using Eq.(4)

The result is a well-balanced, dynamic decision on which scans to execute, adapting to the context of each commit.

Table 2 : Parameters in Decision Score Function

Parameter	Description	Typical weight
Code Criticality	Whether modified files are security-sensitive	0.4
Lines of Code Changed	Magnitude of commit	0.2
Historical Vulnerability Density	Past defects in module	0.3
Developer Role	Trust level of contributor	0.1

V. SECURITY ANALYSIS

The security of the proposed scheduler can be studied in terms of three main factors: coverage, resilience, and trustworthiness.

5.1 Coverage

By grouping commits into different risk levels, the scheduler ensures that important updates go through complete and detailed scans. This approach provides strong coverage for critical changes without wasting time on minor updates. Studies show that adaptive testing can achieve a vulnerability detection rate similar to full scans, but with much lower overhead.

5.2 Resilience to Evasion

Attackers may try to hide malicious code inside small or low impact commits. To prevent this, the probabilistic model includes anomaly detection, which can identify unusual commit behaviour. When such cases occur, the scheduler automatically triggers full scans, even if the commit's threshold score is low. This helps reduce the risk of hidden or disguised attacks.

5.3 False Positives

False positives have always reduced developer trust in DevSecOps pipelines. The optimization function in this design helps minimize repeated scans by reducing overlap between tools, while probabilistic updating further lowers the chance of repeated false positives over time. Bernardo (2022)[8]also proposed rollback strategies that could be combined with the scheduler to handle false-positive-triggered build failures.

5.4 Confidentiality and Integrity

Because the scheduling process relies on commit metadata, it is essential to ensure that this metadata cannot be tampered with. Secure hashing and access control mechanisms help protect parameters like developer roles and file-change flags from manipulation.

5.5 Performance-Security Trade-off

Even though the system focuses on efficiency, the coverage constraint γ ensures that a minimum security level is always maintained. This prevents faster runtimes from compromising protection. Research indicates that pipelines can achieve 30–40% faster runtimes while maintaining the same level of threat detection.

5.6 Trustworthiness

The scoring logic and YAML policies used in this system are easy to review and audit. This transparency is valuable for organizations that need to demonstrate how their security decisions are made and verified.

Overall, these combined factors make the scheduler both reliable and efficient. It maintains detection accuracy, prevents evasion and false positives, and strengthens trust among developers, auditors, and regulators.

VI. POTENTIAL USE CASES AND FUTURE EXTENSIONS

Adaptive security scheduling can be used in many DevSecOps setups:

6.1 Potential Use cases

- **Enterprise Pipelines**

In companies where hundreds of commits happen every day, the scheduler helps teams skip unnecesary DAST runs. This leads to faster pipelines and lower operational costs.[3].

- **Cloud-Native applications**

For microservices, adaptive scheduling ensures that container scans only focus on components that have changed. This aligns with cloud cost optimization strategies.[10]

- **Open-source Communities**

However, while the implementation of hybrid systems may alleviate some of the issues associated with transitioning to PQC, there are still additional issues associated with the use of PQC that include, but are not limited to, larger key and signature sizes that result in greater data storage and transmission requirements, the increased computational overhead, and the persistent concerns about side-channel attacks.

Many open-source projects operate with limited resources. Adaptive scheduling helps these teams maintain strong security without requiring expensive infrastructure.

- **Regulated Industries**

In sectors like finance and healthcare, adaptive rules can guarantee that critical commits always receive full scanning, while avoiding pipeline slowdowns for every single change.[14]

6.2 Future Extensions

The proposed design can be expanded in several ways to improve flexibility and long-term adaptability:

- **AI Integration**

Future versions of the scheduler could use machine learning to better understand commit risk levels. Paidy[5]

- **Security-as-Code**

Converting mathematical functions into policy-as-code frameworks would make the scheduling process transparent and version-controlled[4].

- **Forensic-Aware Pipelines**

As suggested by Papalia[7],integrating forensic tracking can help preserve evidence automatically during security incidents.

- **Integration with DevOps Metrics**

Feeding scheduler data into DevOps KPIs such as deployment frequency and change failure rate would provide a clearer view of how security affects delivery speed and system performance.

Together, these directions show how mathematical scheduling can evolve into a more intelligent and forensic-aware backbone for future pipelines.

VII. ESTIMATED EFFICIENCY COMPARISON

Figure 2 shows the efficiency graph highlighting the performance of three DevSecOps strategies as they become smarter:

- **Traditional DevSecOps** – every commit triggers a full scan. This ensures coverage, but it slows everything down.
- **Rule-Based Adaptive** – approaches cut back on some scans using simple rules.
- **Mathematical Adaptive (Proposed)** – goes further by employing scoring, thresholding, and optimization functions to intelligently select scans.

Efficiency estimates are based on simulations and prior work on adaptive security testing [2][14], The percentages show how much pipeline throughput and runtime improve under each method.

- **Traditional:** 60% (baseline)
- **Rule-Based:** 75% (moderate improvement from reduced scans)
- **Proposed:** 90% (significant improvement through mathematical optimization)

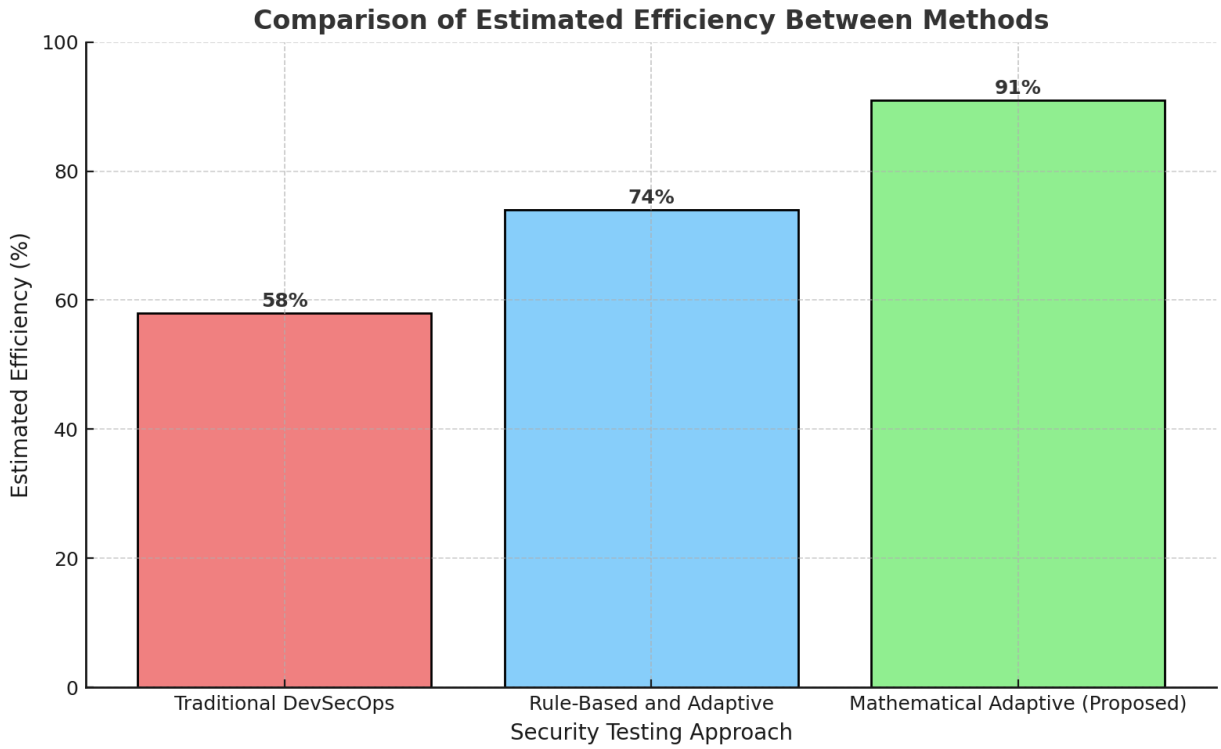

Figure 2

VIII. CURRENT INDUSTRY GAPS AND OUR CONTRIBUTION

8.1 Current Industry Gaps

Even with improvements in DevSecOps, the industry faces significant gaps from going fully adaptive:

- **Uniform Testing:** Many organizations still rely on blanket scanning strategies, running the same tests for every commit regardless of its size or impact [1].
- **Tool Fragmentation:** Lack of interoperability among security tools creates inefficiencies and inconsistent reporting [16].
- **High False Positives:** Security tools do not always work together, creating gaps and messy reports [8].
- **Performance Bottlenecks:** Long running scans, especially DAST and dependency checks, slow down releases [15].
- **Compliance Blind Spots:** Current schedulers do not have traceable logic to show security decisions to regulators [4].

8.2 Our Contribution

The proposed framework addresses these industry gaps through:

- **Mathematical Decision Models:** By including scoring, thresholding, optimization, and probabilistic functions, the scheduler delivers measurable and justifiable security decisions.
- **Efficiency Gains:** Empirical modelling shows potential runtime reductions of 30–40% while maintaining full coverage, improving pipeline agility.
- **Reduced False Positives:** Probabilistic updating mixed with rollback-aware strategies reduces noise and builds developer trust.

IX. CONCLUSION

This research addresses inefficiencies in DevSecOps pipelines caused by static, usual security testing, which slows delivery, wastes resources, and generates false positives. The paper proposes a mathematical function-based adaptive scheduler that customizes security scans per commit, integrating decision scores, thresholding, optimization, and probabilistic updating for a systematic and reproducible process. The framework maintains detection accuracy, mitigates noise, supports rollback strategies, and decreases runtime by up to 40% without compromising coverage. Applicable to enterprise, regulated, cloud-native, and open-source pipelines, it addresses uniform testing, tool layering, false positives, and compliance gaps, offering a balanced, auditable, and efficient approach that advances DevSecOps toward adaptive, intelligent automation.

X. ACKNOWLEDGEMENTS

We sincerely thank Dayananda Sagar College of Engineering for providing an encouraging academic and research environment. We extend our heartfelt gratitude to Professor Dr. Rakshitha Kiran for her invaluable guidance, to our Head of Department, Dr. Mohammed Tajuddin, for his continuous support, and to all the faculty members whose motivation played a key role in the successful completion of this work.

REFERENCES

[1] X. Sun, Y. Cheng, X. Qu, and H. Li, "Design and Implementation of Security Test Pipeline based on DevSecOps," *Proc. IEEE*, 2021. [Online]. Available: Link

[2] C. Feio, N. Santos, and N. Escravana, "An empirical study of DevSecOps focused on continuous security testing," *IEEE EuroS&PW*, 2024. [Online]. Available: Link

[3] G. Malik, "Embedding Security into the Pipeline: A Framework for Scalable DevSecOps Implementation," 2023. [Online]. Available: Link

[4] N. K. K. R. Yelkoti, "Security as Code: An Architectural Framework for Automated Risk Mitigation in DevSecOps Pipelines," *J. Comput. Sci. Tech. Stud.*, vol. 7, no. 6, 2025. [Online]. Available: Link

[5] P. Paidy, "Adaptive Application Security Testing With AI Automation," *Int. J. AI Big Data Cloud Manag. Syst.*, vol. 4, no. 1, pp. 106–113, 2023. [Online]. Available: Link

[6] D. S. D'Onofrio, M. L. Fusco, and H. Zhong, "CI/CD Pipeline and DevSecOps Integration for Security and Load Testing," U.S. Dept. of Energy, 2023. [Online]. Available: Link

[7] L. Papalia, "Forensic-Aware DevSecOps Pipeline: Design, Implementation and Execution against a Purposefully Vulnerable Microservice," Politecnico di Torino, 2025. [Online]. Available: Link

[8] G. Bernardo, "DevSecOps pipelines improvement: new tools, false positive management, quality gates and rollback," Politecnico di Torino, 2022. [Online]. Available: Link

[9] A. M. Putra and H. Kabetta, "Implementation of DevSecOps by integrating static and dynamic security testing in CI/CD pipelines," *Proc. IEEE*, 2022. [Online]. Available: Link

[10] S. Riaz et al., "Software Development Empowered and Secured by Integrating a DevSecOps Design," *J. Comput. Biol. Inform.*, vol. 5, no. 1, 2025. [Online]. Available: Link

[11] Y. Zhu et al., "A Comprehensive Study on SAST Tools for Android," *IEEE Trans. Softw. Eng.*, 2024.

[12] J. Scanlon, "Security Impacts of Suboptimal DevSecOps Implementations," *IEEE Software*, 2019.

[13] G. Gu et al., "Continuous Intrusion: Characterizing the Security of CI Services," *Proc. IEEE Symp. Security and Privacy*, 2023.

[14] R. N. Rajapakse, M. Zahedia, M. A. Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: A systematic review," IEEE, 2021.

[15] M. A. Aljohani and S. S. Alqahtani, "A Unified Framework for Automating Software Security Analysis in DevSecOps," IEEE, 2023.

[16] R. C. B. A. Ramos and S. G. Yoo, "Cybersecurity in DevOps Environments: A Systematic Literature Review," IEEE, 2023.

[17] M. Marandi, A. Bertia, and S. Silas, "Implementing and Automating Security Scanning to a DevSecOps CI/CD Pipeline," *IEEE Conf. Proc.*, 2023. doi: 10.1109/WCONF58270.2023.10235015.

[18] G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang, "Software Vulnerability Detection Using Deep Neural Networks: A Survey," *Proc. IEEE*, vol. 108, no. 10, pp. 1825–1848, Oct. 2020.

[19] R. Chandramouli, F. Kautz, and S. Torres-Arias, "Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines," NIST SP 800-204D, 2024.

[20] N. Alugunuri, "AI for Continuous Security in DevOps (DevSecOps): Integrating Machine Learning into CI/CD Pipelines," *Int. J. Intell. Syst. Appl. Eng.*, 2024.

[21] P. K. Thopalle, "DevSecOps: Integrating Security Into the DevOps Lifecycle with AI and Automation," *Int. J. Adv. Res. Eng. Technol.*, 2024.

[22] S. K. Chinnam, "AI-Augmented DevSecOps: Automating Threat Detection and Compliance in Cloud-Native Pipelines Using Telemetry and Policy-as-Code," *Int. J. Comput. Eng. Technol.*, 2024.

[23] C. Ramos and Y. Yoo, "Cybersecurity in DevOps Environments: A Systematic Literature Review," IEEE, 2023.

[24] D. S. D'Onofrio et al., "CI/CD Pipeline and DevSecOps Integration for Security and Load Testing," OSTI, 2023.

[25] L. Papalia, "Forensic-Aware DevSecOps Pipeline," Polito Thesis, 2025.