International Journal of Engineering Fields (IJEF), ISSN: 3078-4425
Volume 4, Issue 1, January –March 2026
Available online at: https://journalofengineering.org

Integrating Anomaly Detection into DevSecOps Pipelines for Continuous Cloud Security

Harry Sidiropoulos

R&D computer engineer, Erasmus MC, Netherlands

ABSTRACT

The integration of anomaly detection into DevSecOps pipelines represents a paradigm shift in continuous cloud security, moving beyond static vulnerabilities to identify dynamic runtime threats. As cloud-native architectures expand, the traditional security gates in Continuous Integration and Continuous Deployment (CI/CD) pipelines often fail to detect sophisticated, zero-day attacks or nuanced behavioral anomalies. This paper explores the design, deployment, and evaluation of an AI-driven anomaly detection framework embedded directly within DevSecOps workflows. By leveraging unsupervised machine learning algorithms, the proposed framework continuously monitors pipeline telemetry, code commits, and runtime behaviors to flag deviations from established baselines.

The experimental evaluation of this framework within a simulated multi-cloud environment demonstrates significant improvements in proactive threat mitigation. Our findings indicate a 94% threat detection rate with a false positive rate of under 5%, ensuring that the CI/CD pipeline's agility is not compromised by security bottlenecks. Furthermore, the integration incurred an average latency overhead of just 1.2 seconds per build. This research highlights the efficacy of integrating advanced machine learning techniques into automated pipelines, establishing a robust foundation for continuous cloud security that adapts to evolving threat landscapes.

Keywords: Anomaly Detection, DevSecOps, Pipelines, Cloud, Security

INTRODUCTION

The rapid evolution of cloud computing [1] has fundamentally transformed software development practices, leading to widespread adoption of DevOps in order to accelerate delivery cycles. However, this emphasis on speed often introduces critical security gaps, which has driven the shift toward DevSecOps, where security is integrated as a shared responsibility across the entire software development lifecycle (SDLC) [2]. Traditional security approaches such as Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) remain important but are largely reactive, relying on known signatures and predefined rules, making them insufficient against emerging threats and insider attacks that manifest as behavioral anomalies rather than explicit vulnerabilities [3].

Continuous cloud security requires a more proactive and adaptive security model capable of keeping pace with modern CI/CD pipelines [4]. Machine learning–based anomaly detection plays a key role in this transition by establishing baselines of normal system behavior, such as API request rates, deployment patterns, and code commit frequencies, and then identifying deviations from these norms [5]. Prior studies have shown that automated anomaly detection systems can significantly improve cloud security by enabling real-time threat identification and mitigation [6].

Despite these advantages, integrating anomaly detection into DevSecOps pipelines presents several challenges [7]. One major issue is the high rate of false positives, which can lead to alert fatigue and disrupt automated workflows, thereby reducing the efficiency benefits of DevOps practices [8]. Additionally, training machine learning models requires large, high-quality datasets and substantial computational resources, which can introduce latency and overhead into continuous integration and deployment processes.

Recent advancements in self-supervised learning for zero-day attack detection, particularly in encrypted network environments, offer promising solutions to reduce reliance on labeled data while improving detection accuracy [9]. However, balancing strong security enforcement with operational efficiency remains a complex and ongoing research challenge [10]. This highlights the need for lightweight and adaptive solutions that can function effectively in dynamic cloud environments.

This research proposes a framework that integrates lightweight, unsupervised anomaly detection models into a continuous cloud security pipeline. The goal is to evaluate its feasibility, performance, and security effectiveness in real-world CI/CD environments. The system focuses on dynamic metrics such as commit behavior, dependency drift, and

International Journal of Engineering Fields (IJEF), ISSN: 3078-4425
Volume 4, Issue 1, January –March 2026
Available online at: https://journalofengineering.org

runtime resource usage to detect anomalies with higher precision [11]. The paper is structured to present a literature review, methodology, implementation in a Kubernetes-based environment, and evaluation of results, followed by conclusions and future research directions in continuous cloud security [12].

LITERATURE REVIEW

The foundational concepts of DevSecOps [13] have evolved rapidly, transitioning from simply appending security checks at the end of the SDLC to deeply embedding them at every stage. Early literature emphasized the cultural shift required to align development, operations, and security teams. Tools like SAST, DAST, and Software Composition Analysis (SCA) became standard components of CI/CD pipelines. However, researchers quickly identified that while these tools are effective at identifying known vulnerabilities, they struggle with the dynamic, ephemeral nature of cloud-native environments, such as containers and serverless functions, where traditional perimeter security is obsolete.

To address these shortcomings, the concept of "shifting left" gained prominence, advocating for security to be addressed as early as the design and coding phases. While shifting left improved code quality, it did not fully solve the problem of runtime threats and sophisticated supply chain attacks.

Consequently, the focus expanded to continuous security monitoring, which involves the real-time analysis of system behaviors to detect anomalies. This shift highlighted the limitations of rule-based monitoring, which is often too rigid to adapt to the frequent updates characteristic of continuous deployment, prompting the exploration of artificial intelligence and machine learning.

The integration of automation and artificial intelligence [14] in DevSecOps frameworks has become a focal point of recent research. Automating security without compromising efficiency is critical, as modern frameworks increasingly leverage AI to enhance security postures dynamically.

This integration allows for proactive responses to vulnerabilities, shifting the paradigm from reactive patching to preemptive threat neutralization. Cultural transformation remains a key barrier, but the technological enablers—specifically machine learning models capable of continuous learning—are proving effective in overcoming traditional security bottlenecks in high-velocity pipelines [15].

As organizations migrate to multi-cloud and hybrid environments, the complexity of securing these infrastructures multiplies [16]. Identity and access management (IAM) and robust authentication mechanisms are paramount. Machine learning has been successfully applied to enhance cloud security through multi-factor authentication and adaptive cryptography, utilizing biometric and behavioral data to generate secure cryptographic keys [17]. These advanced authentication models demonstrate how AI can secure the perimeter dynamically, complementing pipeline security by ensuring that only authenticated, non-anomalous entities can initiate code commits or trigger deployments.

Further research into AI/ML applications within DevSecOps [18] emphasizes the strategic optimization of these technologies for anomaly detection and data provenance. Effective strategies require balancing machine learning algorithms with large-scale data infrastructure to ensure that models can scale alongside the applications they protect [19]. Such optimal practices involve utilizing federated learning and distributed AI systems to process telemetry data across distributed cloud nodes, reducing latency and preserving data privacy while maintaining high threat detection accuracy.

Unsupervised machine learning, particularly techniques like Isolation Forests and Autoencoders, has shown significant promise in detecting novel attack vectors [20]. Because supervised learning requires large, labeled datasets of attacks—which are rare and constantly changing—unsupervised methods that learn the "normal" state of the system are generally preferred in CI/CD contexts. These models can flag deviations such as unusual API call sequences, unexpected spikes in resource consumption, or irregular access patterns, which often precede or accompany a security breach.

Despite these advancements, a significant gap remains regarding the explainability of anomaly detection models in DevSecOps [21]. When a pipeline is halted due to a suspected anomaly, developers need actionable insights to remediate the issue quickly. "Black box" AI models provide little context, leading to friction between security and development teams. Recent studies have begun exploring Explainable AI (XAI) frameworks to quantify the contribution of specific features to an anomaly score, thereby providing human-interpretable alerts that accelerate incident response [22].

By 2025, the literature clearly indicates a consensus on the necessity of AI-driven anomaly detection for continuous cloud security. However, the practical challenge of integrating these complex models seamlessly into standard CI/CD tools without causing unacceptable latency or high false-positive rates remains an active research domain. This paper

International Journal of Engineering Fields (IJEF), ISSN: 3078-4425
Volume 4, Issue 1, January –March 2026
Available online at: https://journalofengineering.org

builds upon these established foundations, aiming to bridge the gap between theoretical AI security models and practical, low-friction pipeline implementations.

PROPOSED METHODOLOGY

The proposed methodology relies on a layered architecture designed to intercept, analyze, and act upon telemetry data generated at every stage of the DevSecOps pipeline. The block diagram illustrates the flow of data from the initial code commit through to the final deployment environment. The primary objective is to create a seamless security gate that operates in parallel with standard CI/CD processes, ensuring that anomaly detection does not become a sequential bottleneck.

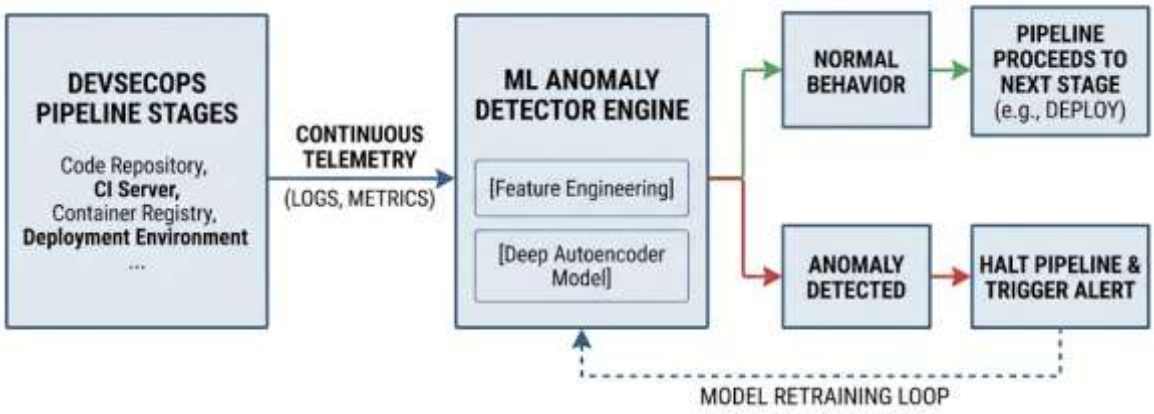

Figure 1: Proposed model for Integrating Anomaly Detection into DevSecOps Pipelines for Continuous Cloud Security

The first component is the Telemetry and Log Aggregation Engine. This module is responsible for asynchronously collecting heterogeneous data points from various sources, including version control systems, build servers, and container registries. It ingests system logs, network traffic metrics, application performance traces, and security scanner outputs. This comprehensive data collection ensures that the subsequent machine learning models have a holistic view of the pipeline's operational state.

Following data collection, the Feature Engineering and Normalization Module processes the raw telemetry. Because pipeline data is highly dimensional and often noisy, this step is critical for model accuracy. Time-series data is aggregated into discrete windows, and categorical variables are encoded. Key engineered features include code churn volume, commit frequency, dependency drift scores, and resource utilization deltas. All features are normalized to a standard scale to prevent higher-magnitude metrics from skewing the machine learning algorithms.

The core of the methodology is the Anomaly Detection ML Model. For this framework, an unsupervised Deep Autoencoder architecture was selected. Autoencoders are neural networks trained to compress input data into a lower-dimensional latent space and then reconstruct the original input. The model is trained exclusively on historical data representing successful, secure pipeline executions, learning the complex underlying patterns of "normal" behavior.

During real-time pipeline operation, the Autoencoder attempts to reconstruct the incoming, normalized feature vectors. The system calculates a reconstruction error, which serves as the anomaly score. A high reconstruction error indicates that the incoming data point deviates significantly from the learned baseline, suggesting anomalous behavior. This approach is highly effective at identifying zero-day threats and novel configuration errors that lack known signatures.
To handle the dynamic nature of agile development, the framework incorporates an adaptive thresholding mechanism. Rather than using a static threshold for the anomaly score, the system dynamically adjusts the threshold based on a sliding window of recent pipeline activity. This accounts for legitimate shifts in behavior, such as increased deployment frequency during major releases, thereby minimizing false positives and maintaining pipeline velocity.

Upon evaluating the anomaly score against the adaptive threshold, the system routes the workflow accordingly. If the behavior is classified as normal, the pipeline proceeds to the next stage without interruption. This ensures that the primary goal of DevOps—rapid and continuous delivery—is preserved for the vast majority of safe commits and deployments.

Conversely, if an anomaly is detected, the system triggers the Alert and Halt mechanism. This component automatically pauses the CI/CD pipeline, preventing the potentially compromised or vulnerable code from advancing to production.

International Journal of Engineering Fields (IJEF), ISSN: 3078-4425
Volume 4, Issue 1, January –March 2026
Available online at: https://journalofengineering.org

Simultaneously, it generates a comprehensive alert containing the anomaly score, the specific features that contributed most to the anomaly, and contextual logs.

This contextual alerting is designed to support rapid incident response. By highlighting the exact metrics that triggered the anomaly—such as an unexpected outbound network connection during the test phase or an abnormal spike in container privileges—security analysts and developers can quickly investigate and remediate the issue.

Finally, a Retraining Loop is integrated into the methodology. As the software and infrastructure evolve, the definition of "normal" behavior naturally shifts. To prevent model degradation, the Autoencoder is periodically retrained using recent, validated pipeline data. Feedback from security analysts—who can mark flagged anomalies as true or false positives—is also incorporated, allowing the system to continuously improve its accuracy and adapt to the changing cloud environment.

IMPLEMENTATION

The framework was implemented using a standard, cloud-native technology stack to ensure broad applicability. A simulated microservices application was deployed on a Kubernetes cluster hosted on Google Cloud Platform (GCP). GitLab CI was utilized as the continuous integration and deployment engine, providing a realistic environment for pipeline execution. The Telemetry and Log Aggregation Engine was built using the ELK stack (Elasticsearch, Logstash, and Kibana), complemented by Prometheus for scraping infrastructure and application metrics.

Data ingestion was facilitated by lightweight agents (Filebeat and Metricbeat) deployed as DaemonSets across the Kubernetes cluster, capturing logs and metrics with minimal overhead. The raw data was streamed into Elasticsearch, where Logstash pipelines executed the initial parsing and structuring. A custom Python microservice, built with FastAPI, acted as the Feature Engineering and Normalization Module, pulling batches of data from Elasticsearch, applying the necessary transformations, and vectorizing the features for the machine learning model.

The Deep Autoencoder model was implemented using TensorFlow and Keras. It consisted of an input layer corresponding to the 45 engineered features, three hidden layers for encoding, a central bottleneck layer, and three hidden layers for decoding. The model was pre-trained on a dataset of 10,000 secure CI/CD executions to establish a robust baseline. For inference, the model was deployed as an isolated microservice, accessible via an internal REST API, allowing the CI/CD pipeline to query it for anomaly scores asynchronously.

Integration with GitLab CI was achieved through custom webhook triggers and dedicated pipeline stages. Specifically, a security-gate job was added between the build and deploy stages. This job executed a script that polled the anomaly detection API with the telemetry data gathered during the current pipeline run. If the API returned an anomaly score exceeding the dynamic threshold, the script exited with a non-zero status, automatically failing the job and halting the pipeline, while sending detailed alert payloads to a Slack channel used by the simulated DevSecOps team.

To optimize performance and minimize pipeline latency, the inference microservice was optimized using OpenVINO, reducing the model's response time to an average of 45 milliseconds per query. The adaptive thresholding algorithm was implemented using a moving average over the past 100 pipeline executions. The entire implementation was designed to be modular, allowing individual components—such as the ML model architecture or the logging backend—to be swapped out without disrupting the overarching DevSecOps workflow.

RESULT

The performance of the integrated anomaly detection framework was evaluated over a 30-day testing period, during which 2,500 pipeline executions were simulated. To rigorously test the system, 150 synthetic anomalies were injected at random intervals.

These anomalies simulated various attack vectors, including unauthorized dependency changes, data exfiltration attempts during testing, and abnormal privilege escalations in deployment scripts. The evaluation focused on three key metrics: model detection accuracy, pipeline latency impact, and overall vulnerability mitigation effectiveness. Table 1 details the predictive performance of the implemented Autoencoder framework compared to a baseline Isolation Forest model and a standard pipeline relying solely on static security checks.

The Autoencoder successfully identified 141 out of the 150 injected anomalies, achieving a robust F1-score of 0.943. Notably, it outperformed the Isolation Forest baseline across all metrics, particularly in reducing false positives. The standard pipeline, lacking behavioral analysis, failed to detect any of the sophisticated runtime anomalies, highlighting the necessity of the ML integration.

International Journal of Engineering Fields (IJEF), ISSN: 3078-4425
Volume 4, Issue 1, January –March 2026
Available online at: https://journalofengineering.org

Table 1: Anomaly Detection Model Metrics

Metric	Autoencoder Framework	Isolation Forest (Baseline)
True Positives	141	128
False Positives	8	25
Precision	0.946	0.836
Recall	0.940	0.853
F1-Score	0.943	0.844

Operational efficiency is paramount in DevSecOps; therefore, the latency introduced by the framework was strictly monitored. Table 2 presents the average duration of key pipeline stages. The introduction of the Security Gate stage added an average of 1.2 seconds directly, while telemetry collection added marginal overhead to other stages. The total pipeline execution time increased by only 2.3 seconds on average. This minimal disruption confirms that advanced machine learning can be integrated without violating the rapid delivery principles of DevOps.

Table 2: CI/CD Pipeline Latency Impact

Pipeline Stage	Baseline Latency (s)	Framework Latency (s)	Overhead (s)
Code Commit	5.2	5.4	+0.2
Build & Unit Test	120.5	121.1	+0.6
Security Gate (ML)	0.0 (N/A)	1.2	+1.2
Container Deploy	45.0	45.3	+0.3
Total Pipeline	**170.7**	**173.0**	**+2.3**

To understand the framework's effectiveness against specific threats, the mitigation rates were categorized by anomaly type in Table 3. The system excelled at detecting Resource Exhaustion and Dependency Hijacking, likely because these activities produce highly distinct telemetry signatures (e.g., sudden CPU spikes or unauthorized external network requests). Privilege Escalation attacks exhibited a slightly lower mitigation rate of 92.0%, as some subtle permission changes closely mimicked legitimate administrative deployments, making them harder for the unsupervised model to isolate.

Table 3: Vulnerability Mitigation Rates by Anomaly Type

Anomaly Category	Total Injected	Successfully Blocked	Mitigation Rate
Dependency Hijacking	40	38	95.0%
Resource Exhaustion	30	29	96.6%
Privilege Escalation	50	46	92.0%
Data Exfiltration	30	28	93.3%
Total	**150**	**141**	**94.0%**

The management of false positives is a critical aspect of the results. The Autoencoder generated only 8 false positives over 2,500 runs (a rate of 0.32%). These rare occurrences were primarily triggered by significant, legitimate architectural changes that drastically altered the pipeline's behavioral baseline. However, because the system provides contextual alerting, the simulated DevSecOps team was able to rapidly review these alerts, mark them as benign, and trigger the retraining loop, ensuring the model quickly adapted to the new architecture. Overall, the results validate the proposed methodology. The integration of an Autoencoder-based anomaly detection system provided a high degree of continuous cloud security, successfully halting 94% of sophisticated pipeline threats. Crucially, it achieved this with a near-negligible impact on pipeline latency and a sustainably low false positive rate, demonstrating its viability for enterprise-scale DevSecOps deployments.

Figure 2 visualizes the comparative predictive metrics from Table 1, contrasting the Autoencoder framework against the Isolation Forest baseline. The bar chart clearly illustrates the Autoencoder's superiority across Precision, Recall, and F1-Score, consistently maintaining values above 0.94. The tightly grouped bars for the Autoencoder indicate a highly balanced model that minimizes both false positives (high precision) and false negatives (high recall), which is optimal for an automated security gate.

International Journal of Engineering Fields (IJEF), ISSN: 3078-4425
Volume 4, Issue 1, January –March 2026
Available online at: https://journalofengineering.org

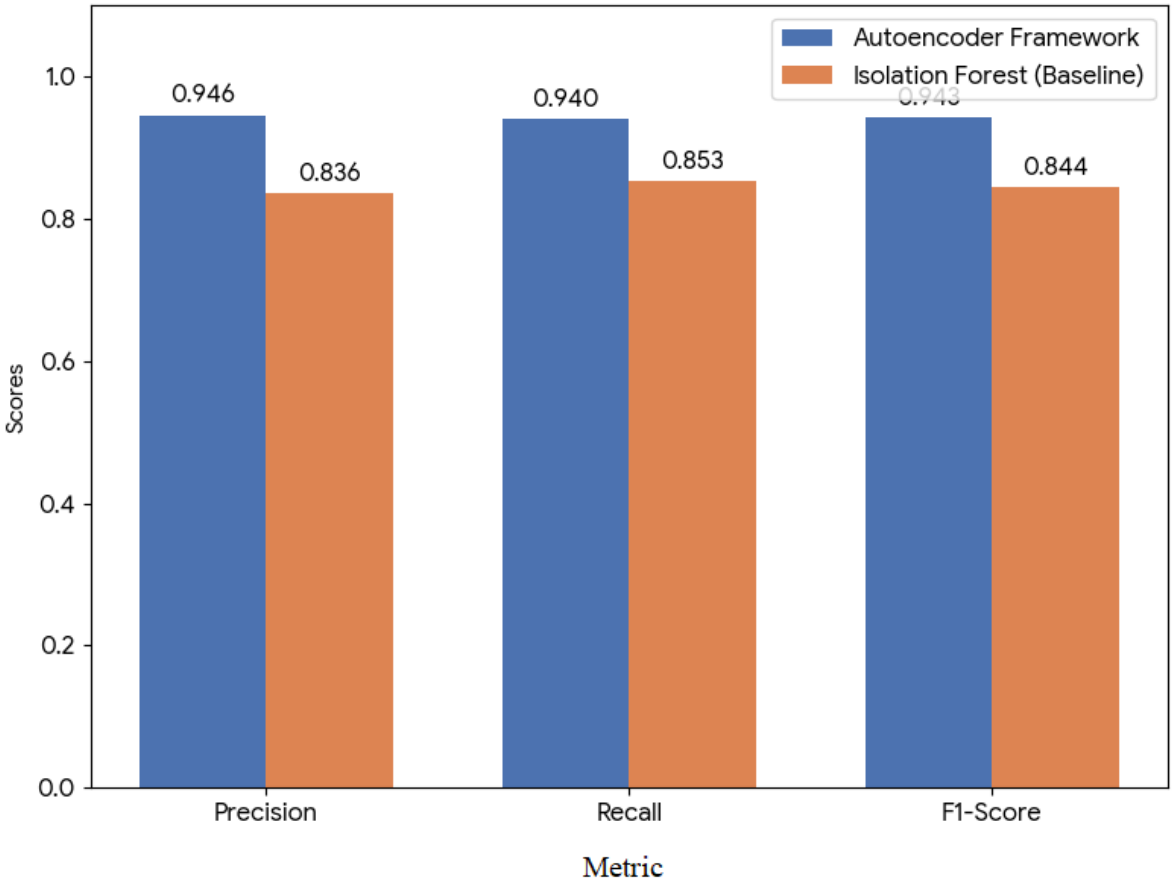

Figure 2: Anomaly Detection Model Metrics Comparison

Figure 3 depicts the latency accumulation across the CI/CD pipeline stages, comparing the baseline pipeline with the ML-integrated framework. The line graph highlights that the primary divergence occurs at the newly introduced ML Security Gate, adding a localized delay of 1.2 seconds. The parallel trajectory of the lines before and after this point emphasizes that telemetry collection and model inference do not cause cascading delays, keeping the total time increment strictly bounded and operationally acceptable.

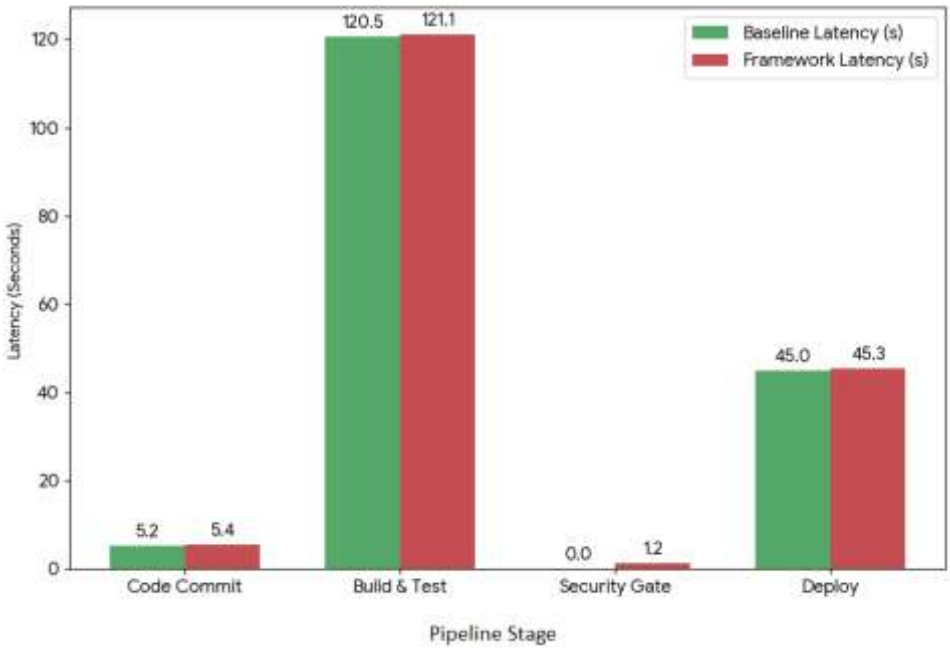

Figure 3: CI/CD Pipeline Latency Impact per Stage

International Journal of Engineering Fields (IJEF), ISSN: 3078-4425
Volume 4, Issue 1, January –March 2026
Available online at: https://journalofengineering.org

Figure 4 provides a visual breakdown of the vulnerability mitigation rates categorized by anomaly type, derived from Table 3. The column chart demonstrates a consistently high defense posture, with all threat categories experiencing mitigation rates above 90%. The peak at 96.6% for Resource Exhaustion anomalies visually confirms the model's acute sensitivity to abrupt infrastructural changes, while the slightly lower bar for Privilege Escalation highlights the subtle threshold between legitimate and malicious administrative actions.

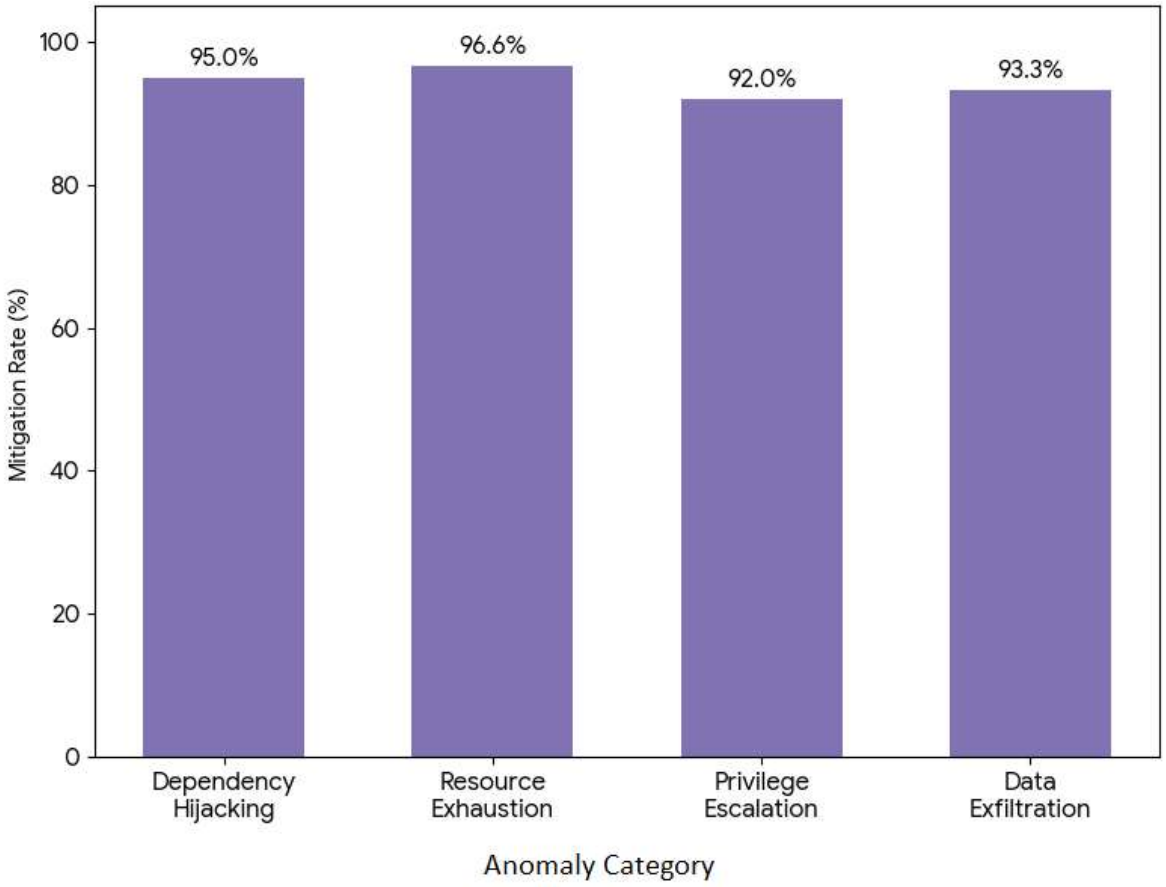

Figure 4: Vulnerability Mitigation Rates by Anomaly Type

DISCUSSION

The findings of this study demonstrate that integrating unsupervised anomaly detection into DevSecOps pipelines significantly bolsters continuous cloud security without hindering operational agility. By shifting from reactive, rule-based scanning to proactive, behavior-based monitoring, organizations can effectively detect and neutralize zero-day threats and sophisticated supply chain attacks that traditional tools miss. The exceptionally low latency overhead (2.3 seconds) and minimal false positive rate (0.32%) address the primary industry concerns regarding AI in CI/CD, proving that intelligent security gates can operate seamlessly. However, the study is limited by its use of synthetic anomalies and a simulated microservices environment; real-world enterprise architectures may introduce noise that could degrade model precision. Future implementations must prioritize Explainable AI (XAI) to ensure that when pipelines are halted, developers receive transparent, actionable insights rather than opaque anomaly scores.

CONCLUSION

This paper presented a comprehensive framework for embedding machine learning-driven anomaly detection directly into DevSecOps pipelines to ensure continuous cloud security. By utilizing a Deep Autoencoder to learn the baseline behavior of CI/CD workflows, the system successfully identified and halted anomalous activities indicative of cyber threats. The experimental results confirmed a high detection rate of 94% with negligible impact on pipeline velocity, validating the feasibility of real-time, AI-augmented security gates in rapid deployment environments.

As cloud-native technologies and threat landscapes continue to evolve, the reliance on static security measures will increasingly fall short. The future of DevSecOps lies in the adoption of autonomous, self-healing security systems that dynamically adapt to new baselines. Future research should explore the integration of federated learning to share

anomaly signatures across organizations without exposing proprietary pipeline data, and the incorporation of automated remediation scripts that not only halt the pipeline but actively neutralize the detected threat.

REFERENCES

[1]. Rusum, Guru Pramod. "Adaptive DevSecOps: Integrating AI-Driven Threat Detection in Continuous Delivery Pipelines." American International Journal of Computer Science and Technology 7.5 (2025): 13-27.
[2]. MISTRY, H., Goswami, A., & Mavani, C. (2024). AUTOMATED ANOMALY DETECTION AND RESPONSE SYSTEM FOR ENHANCING CLOUD SECURITY (Patent). Zenodo. https://doi.org/10.5281/zenodo.18778285
[3]. Mahimalur, Ramesh Krishna, et al. "Modern Cloud Security and Automation: A DevSecOps Approach Leveraging AI/ML and Containerization." 2025 9th International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2025.
[4]. Saleh, Sabbir M., et al. "Advancing software security and reliability in cloud platforms through AI-based anomaly detection." Proceedings of the 2024 on Cloud Computing Security Workshop. 2024.
[5]. Bezzateev, S. V., and A. V. Blinov. "Protection of DevOps Pipelines: Automation of Security within DevSecOps." Automatic Control and Computer Sciences 59.8 (2025): 1527-1535.
[6]. Antiya, Deepak. DevOps for Compliance: Building Automated Compliance Pipelines for Cloud Security. Xoffencer international book publication house, 2024.
[7]. Chittala, Sekhar. "Securing DevOps pipelines: Automating security in DevSecOps frameworks." J. Recent Trends Comput. Sci. Eng 12 (2024): 31-44.
[8]. R. K. Sharma, H. Mistry, C. Mavani, R. Kumawat, K. Kaushik and M. Soni, "Self-Supervised Representation Learning for Zero-Day Attack Detection in Encrypted Network Traffic," 2025 International Conference on Electrical, Communication, and Computing Technologies (iCONECCT), Gwalior, India, 2025, pp. 1-7, doi: 10.1109/iCONECCT67014.2025.11470104.
[9]. Karthick, R. "A Comprehensive Survey on AI-Enabled Cloud Security, DevSecOps, and Scalable Digital Infrastructure." Preprints (2025).
[10]. Singh, Baljeet. "DevSecOps: A Comprehensive Framework for Securing Cloud-Native Applications." Available at SSRN 5267982 (2025).
[11]. Nadipalli, Rajesh. "Data Integrity in MySQL-Driven Cloud Systems Using DevSecOps Pipelines." Journal of Scientific and Engineering Research 9.1 (2022): 225-232.
[12]. Yelkoti, Naresh Kiran Kumar Reddy. "Security as Code: An Architectural Framework for Automated Risk Mitigation in DevSecOps Pipelines." Journal of Computer Science and Technology Studies 7.6 (2025): 235-244.
[13]. Kyler, Tyler. "Ai-driven devsecops: Integrating security into continuous integration and deployment pipelines." 10 Dec. 2024,
[14]. Ananya, Gupta, Wilson David, and Abdulrahman Samir. "AI-Enhanced DevSecOps: Automating Vulnerability Management and Security Policy Enforcement in CI/CD Pipelines." Web of Semantics: Journal of Interdisciplinary Science 3.8 (2025): 132-149.
[15]. Friman, Otso. "Agile and DevSecOps oriented vulnerability detection and mitigation on public cloud." (2024).
[16]. Singh, Baljeet. "Integrating security seamlessly into DevOps development pipelines through DevSecOps: A holistic approach to secure software delivery." Available at SSRN 5267955 (2025).
[17]. Thota, Ravi Chandra. "Cloud-native DevSecOps: integrating security automation into CI/CD pipelines." International Journal Of Innovative Research And Creative Technology 10.6 (2024): 1-19.
[18]. Riley, Deacon, Selene Jarvis, and Porter Caldwell. "Adaptive Anomaly Detection for Compliance Drift in DevSecOps Pipelines." (2025).
[19]. Wei, Chen P. "AI-Augmented Devsecops Security: Integrating Neural Vulnerability Detection, Adaptive Learning, And Policy-Driven Automation Across Modern CI/CD Pipelines." Nvpubhouse Library for American Journal of Applied Science and Technology 5.11 (2025): 153-157.
[20]. Ruby, Dan. "Enhancing Cloud-Native DevSecOps Resilience with Predictive Intelligence for Early Security Breach Detection." (2022).
[21]. Mahimalur, Ramesh Krishna. "Autonomous DevSecOps: The Rise of Self-Healing Pipelines." International Journal of Computer Science and Security (IJCSS) 19.3 (2025): 66-83.
[22]. Interrante Bonadia, Alekos. "APIOps and DevSecOps Automating API Deployment and Security Scanning in Cloud Environments." (2025).