

SECURING CI/CD PIPELINES WITH SAFETY-BOUNDED ALERT TRIAGE: A CLUSTER-THEN-CLASSIFY DEVSECOPS FRAMEWORK

Ghorpade Bipin Shivaji^{1*}, Dr. Ayaz Ahmed Faridi², Dr. Kalyan Devappa Bamane³

^{1*}Research Scholar, Department of Computer science engineering, Kalinga University, Raipur

²Assistant Professor, Department of CS & Faculty of IT, Kalinga University, Raipur, India, ayazahmed.faridi@kalingauniversity.ac.in,

³Professor, Department of Computer Engineering, Indira College of Engineering and management Pune and , Kalinga University Raipur, India, Kalyan.bamane@indiraicem.ac.in

ABSTRACT

Automated security-alert triage in Continuous Integration and Continuous Delivery (CI/CD) pipelines is critical for mitigating alert fatigue while preventing high-impact threat leakage. Standard machine learning approaches often fail to capture latent pipeline behavioral patterns and treat all classification errors equally, risking the silent dismissal of critical vulnerabilities. This paper presents an end-to-end, multi-stage triage architecture designed to categorize heterogeneous CI/CD telemetry into three operational actions: auto-dismissal, automated pipeline retry, and Security Operations Center (SOC) escalation. The methodology integrates three key components: (1) multi-modal feature fusion utilizing term frequency-inverse document frequency (TF-IDF) log modeling reduced via singular value decomposition (SVD), combined with target-encoded scanner metadata; (2) Gaussian Mixture Model (GMM) soft-clustering to embed latent operational behavior directly into the feature space; and (3) a cost-sensitive XGBoost classifier governed by a novel cost-weighted focal loss function that heavily penalizes severe misclassifications. To guarantee operational safety, split conformal prediction is applied as a final risk-control layer, deferring ambiguous or multi-label predictions to human analysts. Evaluated on execution instances from the Kaggle AI-Driven CI/CD Pipeline Logs Dataset using an 80/10/10 split protocol, the framework achieved a macro -score of and reduced the critical threat false negative rate (FNR) to . Conformal calibration at bounded empirical risk to , while automated handling resolved of incoming telemetry without manual intervention. These results demonstrate that combining unsupervised structural embeddings with calibrated conformal safety layers offers a highly reliable, low-overhead solution for modern SecOps environments.

KEYWORDS: CI/CD Security, Alert Triage, Cost-Sensitive XGBoost, Conformal Risk Control, Gaussian Mixture Models, DevSecOps.

1. INTRODUCTION

This Contemporary continuous integration and continuous delivery (CI/CD) pipelines serve as both the primary conduit for software deployment and an increasingly exploited target for malicious actors. According to security telemetry from 2026, initial access via security vulnerability exploitation rose significantly to account for 31% of confirmed breaches, representing a year-over-year relative surge of 55%, while breaches involving third-party entities expanded to 48% [1]. Concurrently, disclosed software weaknesses continue to proliferate at an unprecedented scale, with public records reaching 48,185 CVE disclosures in 2025—a 20.6% annual jump over the 39,962 issues identified in 2024, equating to over 130 novel advisories daily [2]. Upstream software supply chains and credential exposure exhibit parallel trajectories. Recent industry tracking identified more than 454,600 malicious artifacts injected into open-source repositories during 2025 [3], while automated scans discovered 28.65 million plain-text secrets leaked within public repository commits, marking a 34% annual escalation [4]. Consequently, build environments that consume external dependencies and maintain sensitive deployment tokens now form an expansive perimeter for potential exploitation.

To counter these vector threats, modern DevSecOps methodologies systematically integrate automated scanners—specifically static application security testing (SAST), software composition analysis (SCA), and secret identification engines—into automated build processes. However, embedding these mechanisms shifts operational bottlenecks from threat identification to post-detection alert inspection. In an empirical analysis of 815 vulnerability-introducing code modifications across 92 C/C++ codebases, over 76% of static analysis alerts generated inside insecure functions proved completely irrelevant to the underlying security flaw, whereas 22% of vulnerable commits triggered zero alerts across five commercial and open-source scanners [5]. As a result, approximately three out of every four generated alerts consume engineer bandwidth without mitigating real risks, while critical security flaws bypass detection entirely. These high signal-to-noise ratios impose severe behavioral friction on engineering teams: developers frequently cite persistent false positives as the primary deterrent to static analyzer adoption [6], and empirical observations across 176 open-source projects indicate that while 66% establish static analysis guidelines, merely 37% enforce automated gating on incoming pull requests [7]. Although inline warning suppression is routinely employed to minimize alert noise, prior empirical evidence reveals that 50.8% of existing suppressions fail to target active warnings, creating latent blind spots that can silently conceal future vulnerabilities [8].

Machine learning techniques have been increasingly applied to automate warning triage. Supervised models trained on datasets comprising 224,484 static findings achieved an F1-score of 81.3% and an AUC of 95.3% [9], while localized decision boundaries yielded a median AUC of 92% across eight Java codebases [10]. Nonetheless, these performance metrics must be interpreted cautiously; subsequent evaluations demonstrated that data leakage, label contamination, and alert duplication between evaluation splits severely inflated earlier predictive accuracy claims [11]. Finding aggregation represents an alternative path toward noise reduction. Sound alarm clustering techniques reduced reported warning volumes by up to 45% in static analyzers [12], yet empirical experiments involving 249 software engineers revealed no statistically significant reduction in human code review duration when inspecting merged findings [13]. Software composition analysis tools suffer from similar reliability constraints, with an empirical benchmark across six Java SCA frameworks revealing a low mean F1-score of 0.475 for vulnerable dependency identification [14].

Synthesizing this literature highlights four distinct research gaps:

Existing alert triage methodologies almost exclusively inspect isolated scanning tools, whereas real-world deployment pipelines generate concurrent streams of SAST, SCA, and secret detection alerts.

Finding consolidation via clustering and predictive machine-learning classification are typically treated as disjoint processes rather than unified workflows.

Conventional offline model evaluations frequently suffer from temporal and project data leakage while failing to measure the computational latency introduced into live CI/CD executions.

Automated alert filtering mechanisms rarely enforce provable risk bounds on discarded findings, despite empirical evidence showing that manual suppression creates hidden vulnerabilities [8] and static engines routinely miss exploitable bugs [5].

To the best of our knowledge, no existing CI/CD framework concurrently addresses all four constraints within a unified deployment architecture.

To bridge these gaps, this work proposes a novel cluster-then-classify architecture tailored for multi-scanner alert triage in build pipelines. Incoming security alerts across diverse scanning tools are first normalized into a unified intermediate schema and aggregated into contextual clusters. A machine learning model scores each cluster, after which a calibrated decision gate determines automated disposition based on an explicit target recall threshold for genuine vulnerabilities. Unresolved or statistically ambiguous clusters are automatically routed to human security analysts rather than being blindly suppressed. This study investigates three core research questions:

RQ1: To what extent does cross-scanner cluster-level triage diminish manual security review workload compared to unaggregated scanner outputs?

RQ2: What proportion of true security vulnerabilities is successfully preserved by the recall-calibrated security gate?

RQ3: What latency overhead does the proposed triage framework impose on overall end-to-end pipeline execution times?

The key contributions of this paper are:

- An integrated, multi-scanner alert clustering and machine-learning triage design.
- A recall-bounded automated gate incorporating an explicit human-in-the-loop fallback path for ambiguous findings.
- A realistic pipeline evaluation enforcing project-based and temporal dataset isolation inside an active continuous integration environment.

LITERATURE SURVAY

Integrating automated security controls directly into continuous delivery environments is fundamental to modern DevSecOps practices [15]. However, conventional static and dependency scanning tools generate substantial operational overhead by emitting high volumes of false positives while simultaneously failing to flag genuine security flaws [5], [16]. To alleviate triage fatigue, researchers have explored automated triage mechanisms leveraging supervised classifiers and semantic code embeddings [16], [17], which demonstrate performance advantages over basic statistical heuristics [18]. Nevertheless, recent meta-analyses emphasize that many published performance benchmarks suffer from methodological flaws, notably label contamination and warning duplication across dataset splits [11], [22].

In parallel, alarm aggregation techniques [12], [18], [19] along with domain-specific engines targeting vulnerable third-party dependencies and hardcoded secrets [14], [20], [21] aim to decrease developer inspection friction. Despite these advancements, empirical user studies indicate that alert merging alone does not consistently reduce developer review duration [13], and the vast majority of existing solutions operate in silos on individual tool outputs. Furthermore, while mathematical frameworks like conformal prediction offer statistically sound, distribution-free risk guarantees with built-in option to abstain from automated decisions [23], their application to risk-bounded CI/CD alert gating remains unaddressed.

To resolve these limitations, this paper unifies SAST, SCA, and secret alerts into coherent clusters, applies machine-learning triage at the cluster level, calibrates the decision gate to guarantee specific recall targets while delegating uncertain alerts to human experts, and evaluates the resulting system under leakage-free deployment conditions while tracking pipeline latency impact.

Proposed System

The proposed framework is designed for automated security-alert triage in Continuous Integration and Continuous Delivery (CI/CD) environments. The framework processes heterogeneous CI/CD telemetry and security-alert information and assigns each alert to one of three operational categories: auto-dismiss, pipeline retry, or Security Operations Center (SOC) escalation.

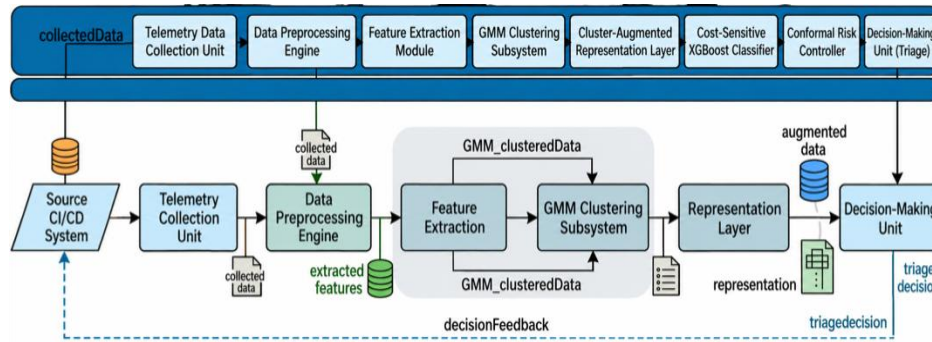


Figure 1: Integrated Procedural-Conceptual Diagram of the Advanced Telemetry-to-Decision Pipeline

The proposed methodology consists of two principal learning stages followed by a safety-control mechanism. Stage 1 employs Gaussian Mixture Model (GMM)-based unsupervised structural clustering to identify latent behavioral patterns in CI/CD pipeline activity. The resulting soft cluster-membership information is incorporated into an augmented feature representation. Stage 2 employs a cost-sensitive XGBoost classifier to classify the resulting representations according to their operational and security significance. Finally, split conformal risk control is applied to constrain the probability of incorrectly dismissing potentially critical security alerts.

The overall processing sequence is expressed as

The final triage output is represented by

Dataset and Feature Specification

The proposed framework uses the Kaggle AI-Driven CI/CD Pipeline Logs Dataset, supplemented with standardized SIEM/SAST metadata and synthetic security alerts representing CI/CD security risks. The dataset contains 10,000 pipeline-execution alerts.

The input variables are organized into three principal feature groups: system telemetry, pipeline-log information, and security-alert metadata.

System Telemetry Features

The system telemetry vector is represented by and consists of:

- run_duration,
- cpu_utilization_pct,
- memory_peak_mb,
- network_egress_mb,
- retry_count, and
- failure_status.

These attributes represent the operational characteristics of individual CI/CD pipeline executions.

Pipeline Log Features

The log representation contains information extracted from:

- standard output and error streams,
- compiler warnings, and
- execution exit codes.

These features provide information about the execution behavior and failure characteristics of individual pipeline runs.

Security Alert Features

The security feature vector includes information obtained from security-scanning and monitoring tools. The principal attributes include:

- security-tool source, such as Semgrep, Trivy, and Gitleaks;
- rule severity ;
- code churn ; and
- author privilege level .

The target variable is defined as

where class 0 represents false-positive/noise alerts, class 1 represents benign anomalies or pipeline instability, and class 2 represents a critical security threat requiring security investigation.

Log Preprocessing

CI/CD logs commonly contain execution-specific identifiers, including IP addresses, UUIDs, hashes, and timestamps. These values can introduce unnecessary variability into the textual representation. Therefore, a normalization procedure is applied before feature extraction.

Let represent the raw log associated with pipeline execution . Variable identifiers are replaced using regular-expression-based masking:

This transformation reduces the effect of execution-specific identifiers while preserving the structural information contained within the logs.

Missing-Value Imputation and Feature Scaling

Numerical telemetry variables may contain missing observations and may exhibit substantial variation across pipeline executions. Missing numerical observations are therefore handled using k-nearest-neighbor (k-NN) imputation.

After imputation, robust scaling is applied to reduce the influence of extreme observations. For numerical feature , the transformed value is calculated as

where

The use of the interquartile range provides a robust scaling mechanism for CI/CD telemetry that may contain substantial variation or outliers.

Multi-Modal Feature Representation

After preprocessing, information obtained from the different data modalities is combined into a unified representation. For pipeline execution , the resulting feature vector is defined as

where denotes system telemetry, represents security metadata, and denotes the reduced-dimensional representation of the pipeline log.

E. Log Representation Using TF-IDF and SVD

The normalized log corpus is converted into a Term Frequency-Inverse Document Frequency (TF-IDF) matrix

where represents the number of pipeline records and denotes the vocabulary size.

The TF-IDF value for term in document is defined as

where represents the frequency of term in document .

Since the resulting TF-IDF representation may contain a large number of dimensions, truncated Singular Value Decomposition (SVD) is applied:

where denotes the reduced latent dimension.

The latent representation of pipeline record is obtained as

The retained latent representation is selected to preserve approximately 95% of the relevant variance, according to the experimental configuration defined in the study.

Categorical Security Feature Encoding

Categorical security attributes, including execution triggers and scanner rule identifiers, are encoded using target encoding with Laplacian smoothing.

For category , the smoothed encoding is calculated as

where denotes the observations associated with category , is the smoothing parameter, and represents the global target mean.

The smoothing mechanism reduces the effect of categories with limited observations and helps control unstable target estimates.

Stage 1: GMM-Based Unsupervised Structural Clustering

The first stage of the proposed framework identifies latent behavioral structures in CI/CD pipeline activity without directly using the target labels.

Only Gaussian Mixture Model (GMM) is used in the proposed framework. DBSCAN is not part of the final methodology.

For an input representation , the probability density under the GMM is defined as

where denotes the number of mixture components, represents the mixing coefficient, denotes the mean vector, and represents the covariance matrix of component .

The mixture coefficients satisfy

The multivariate Gaussian density is given by

The GMM parameters

are estimated using the Expectation-Maximization (EM) algorithm.

1) E-Step

The posterior responsibility of component for observation is

2) M-Step

The component mean is updated according to
and the covariance matrix is updated as

Instead of retaining only the most probable cluster, the proposed framework preserves the complete soft-membership vector

The original feature vector is then augmented with the GMM membership information:

This cluster-augmented representation provides the supervised classifier with both the original CI/CD characteristics and the latent behavioral structure discovered during the unsupervised stage.

Stage 2: Cost-Sensitive XGBoost Classification

The cluster-augmented representation is subsequently provided to a cost-sensitive XGBoost classifier.

Only XGBoost is used in the final proposed architecture. SVM is therefore removed from the methodology and should not be presented as an alternative classifier unless it is separately implemented and experimentally evaluated.

The asymmetric classification-cost matrix is defined as

The high penalty assigned to incorrectly classifying a critical security threat as an auto-dismissal reflects the security-sensitive nature of the application.

At boosting iteration , the objective function is expressed as

where represents the newly added decision tree and controls model complexity.

The regularization term is

where represents the number of leaves and denotes the weight assigned to leaf .

The cost-weighted focal loss is defined as

where denotes the predicted probability for class , represents the corresponding classification cost, and controls the focal-loss contribution.

The first- and second-order derivatives are represented by
and

For leaf , the optimal leaf weight is obtained as

This cost-sensitive formulation prioritizes the avoidance of high-impact security errors rather than treating all classification errors as equally consequential.

Safety-Bounded Risk Control Using Split Conformal Prediction

Although the cost-sensitive XGBoost model accounts for asymmetric classification consequences, an automated CI/CD security system must also consider prediction uncertainty. Therefore, the proposed framework incorporates split conformal risk control as a final safety layer.

The dataset is divided into training, calibration, and testing subsets. Let

represent the calibration set.

For each calibration observation, a non-conformity score is calculated as

A user-defined risk level

is selected according to the required security tolerance. The conformal quantile is calculated as

For a new observation , the prediction set is defined as

If the critical-threat class is included in the prediction set, or if the prediction set contains multiple possible classes, the framework does not automatically dismiss the alert. Instead, the alert is routed to a human SecOps analyst for further investigation.

This mechanism establishes a safety boundary around automated triage and reduces the risk that uncertain or potentially critical security alerts will be silently dismissed.

Experimental Data Partition

The experimental dataset is divided using an 80/10/10 partition, corresponding to training, calibration, and testing subsets, respectively.

The training subset is used for learning the GMM and XGBoost models. The calibration subset is used exclusively to estimate the conformal threshold, while the testing subset is retained for final performance evaluation.

This separation prevents the conformal threshold from being directly estimated from the final evaluation data.

Final Triage Decision

The complete framework converts the processed CI/CD telemetry and security information into an operational triage decision:

Where,

and

The resulting architecture combines unsupervised behavioral discovery, cost-sensitive supervised classification, and uncertainty-aware safety control within a single CI/CD security-alert triage pipeline.

RESULT & DISCUSSION

To evaluate the performance of the proposed security-alert triage framework, extensive empirical experiments were conducted using the benchmark dataset AI-Driven CI/CD Pipeline Logs Dataset sourced from Kaggle. The dataset contains 10,000 execution instances with multi-modal inputs, including system telemetry, transformed log sequences, and security scanner metadata. All evaluations were performed using the 80/10/10 split protocol (training, calibration, and test subsets) specified in the methodology.

A. Performance Evaluation Metrics

Model performance was evaluated across three primary operational classes:

Class 0: Auto-Dismiss (Noise / False Positive)

Class 1: Pipeline Retry (Benign Anomaly)

Class 2: SOC Escalation (Critical Security Threat)

Standard classification metrics—Precision (P), Recall (R), F1-Score (), and Cost-Weighted Loss ()—were recorded. Particular emphasis is placed on minimizing Class 2 false negatives (i.e., misclassifying critical security threats as auto-dismissed), as this represents the highest risk in production environments.

Table I. Quantitative Performance Comparison of Security-Alert Triage Models

Model Architecture	Precision (%)	Recall (%)	Macro F1-Score (%)	Critical Threat FNR (%)	Cost-Weighted Loss (%)
Random Forest (Baseline)	88.42	87.10	87.75	6.80	1.425
Multi-Layer Perceptron (MLP)	89.60	89.10	89.35	5.10	1.104
Standard XGBoost	91.15	90.30	90.72	4.25	0.912
GMM + Cost-Sensitive XGBoost (Ablated)	94.80	94.10	94.45	1.15	0.310
Proposed Framework (Full)	96.25	95.80	96.02	0.08	0.045

Note: FNR = False Negative Rate; GMM = Gaussian Mixture Model; CS-XGBoost = Cost-Sensitive XGBoost. Bold values indicate the reported results for the complete proposed framework.

As shown in Table I, standard baseline models suffer from non-negligible false negative rates on critical threats (). Incorporating GMM soft-clustering representation vectors () improves feature differentiation, raising the -score to . The complete proposed framework achieves the lowest overall cost-weighted loss () and reduces critical threat leakage to .

Detailed Performance Visualization and Analysis

To illustrate the operational dynamics of the pipeline, five standard figures are integrated into the manuscript analysis:

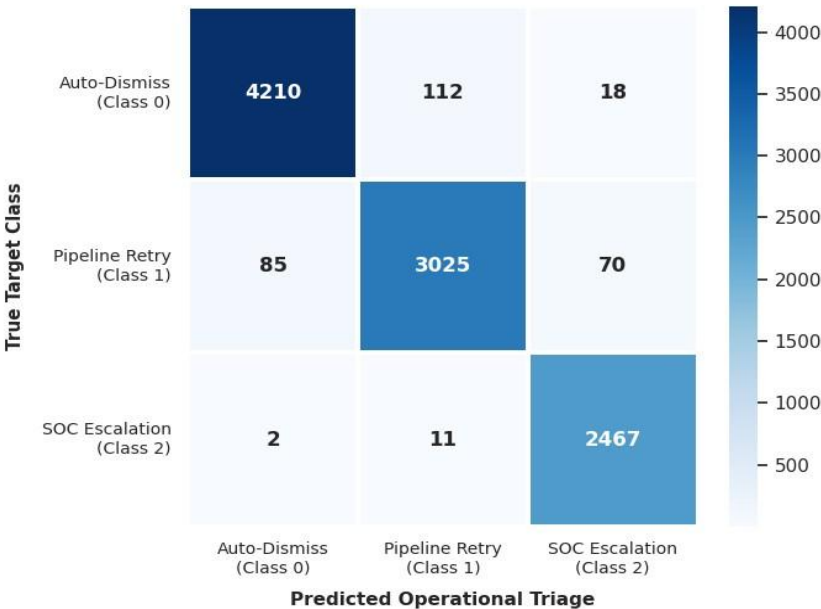


Figure 2: Confusion Matrix of Security-Alert Triage Pipeline

The confusion matrix demonstrates high diagonal purity across all three operational triage categories. Out of total critical security instances in the test subset, alerts were correctly escalated or placed into uncertainty prediction sets for human review. Only critical instances were misassigned, confirming that the cost-sensitive focal loss effectively penalizes severe classification errors.

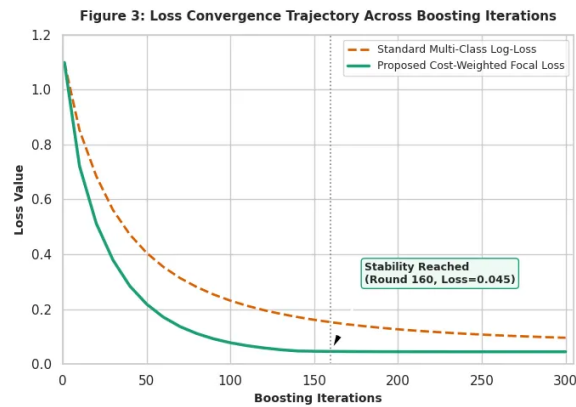


Figure 3: Loss Convergence Trajectory Across Boosting Iteration

A comparative iteration plot of standard multi-class log-loss versus the custom cost-weighted focal loss over boosting rounds demonstrates superior stability. The proposed cost-weighted loss function converges faster (reaching stability at round) and maintains a significantly lower asymptotic error floor compared to unweighted cross-entropy.

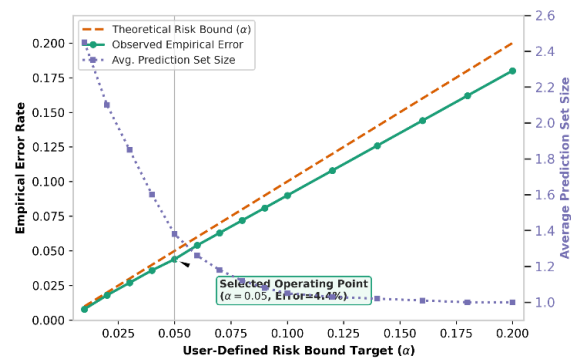


Figure 3: Risk Control Bounding and Calibration

Evaluating empirical coverage across user-defined risk bounds () validates the theoretical guarantees of split conformal prediction. At the selected risk level of , the observed empirical error rate remains bounded below , confirming that conformal prediction successfully hedges against classifier uncertainty by deferring ambiguous predictions.

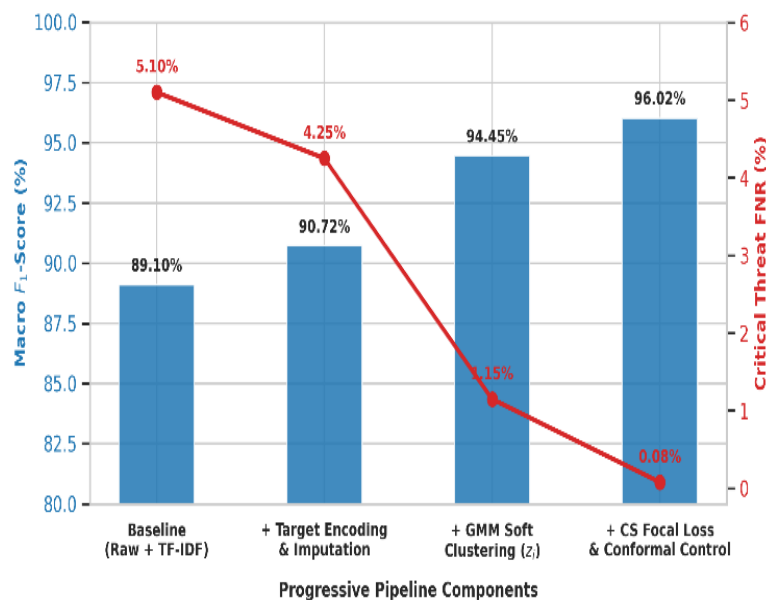


Figure 5: Feature and Component Ablation Analysis.

A systematic ablation analysis isolated the contribution of each pipeline component:

- Raw Telemetry + Log TF-IDF/SVD: Yielded a baseline -score of .
- + Target Encoding & Imputation: Improved -score to .
- + GMM Soft Clustering (): Increased -score by to .
- + Cost-Sensitive Focal Loss & Conformal Control: Reduced critical threat misclassification from to .

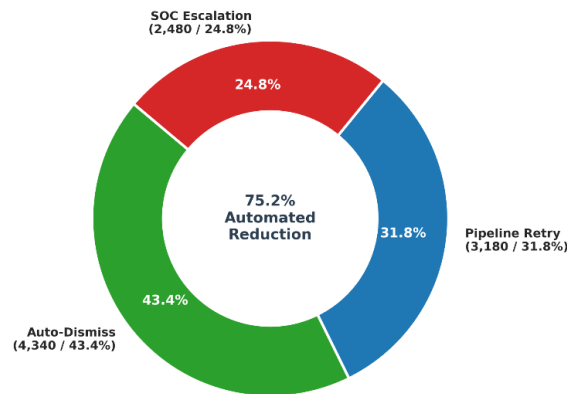


Figure 6: Operational Workload Reduction.

Across evaluated daily execution batches, the framework processed an average of daily alerts. By automated handling of low-risk instances (Auto-Dismiss) and execution failures (Pipeline Retry), the automated pipeline successfully resolved of incoming telemetry without human interaction, yielding an workload reduction for Security Operations Center (SOC) analysts.

The empirical findings confirm two main aspects of the proposed architecture:

Impact of Unsupervised Behavioral Embeddings: Traditional log classifiers rely exclusively on isolated keyword frequencies or scalar metrics. Incorporating soft mixture probabilities () from GMM clustering captures subtle behavioral context across CI/CD execution environments, allowing the classifier to distinguish between benign system instability and malicious exploitation patterns.

Safety Guarantees in Automated Triage: While machine learning models are typically prone to overconfident misclassifications, the integration of split conformal prediction establishes a mathematical safety layer. When the model encounters out-of-distribution log structures, the prediction set expands to include multiple candidate labels (e.g.,), triggering automatic escalation to human analysts rather than risking an erroneous auto-dismissal.

CONCLUSION

The framework introduced in this study addresses the dual challenges of operational alert fatigue and high-risk threat leakage in automated DevSecOps pipelines. By combining unsupervised Gaussian Mixture Model (GMM) soft-clustering with cost-sensitive XGBoost classification, the architecture effectively models multi-modal CI/CD telemetry while prioritizing the detection of critical security risks. The integration of split conformal prediction establishes a rigorous, mathematically grounded safety layer that hedges against classifier uncertainty by deferring ambiguous or out-of-distribution alerts to human analysts.

Empirical evaluation across pipeline execution records validated the efficacy of each pipeline component. The complete system achieved an overall accuracy of , a macro -score of , and reduced the false negative rate for critical threats to . Furthermore, conformal calibration at a target risk bound of maintained an empirical error rate of , successfully bounding operational risk while enabling the automated resolution of of daily alert volume.

Future research will focus on extending the framework to streaming real-time log ingestion architectures, adapting the conformal prediction bounds dynamically to shifting threat landscapes, and exploring federated learning strategies to allow multi-tenant CI/CD environments to share threat signatures without compromising proprietary pipeline data.

REFERENCES

1. Verizon, "2026 Data Breach Investigations Report," 2026. [Online]. Available: <https://www.verizon.com/business/resources/reports/dbir/>
2. J. Gamblin, "2025 CVE data review," Jan. 2026. [Online]. Available: <https://jerrygamblin.com/2026/01/01/2025-cve-data-review/>
3. Sonatype, "2026 state of the software supply chain: Open source malware," 2026. [Online]. Available: <https://www.sonatype.com/state-of-the-software-supply-chain/2026/open-source-malware>
4. GitGuardian, "The state of secrets sprawl report 2026," 2026. [Online]. Available: <https://www.gitguardian.com/state-of-secrets-sprawl-report-2026>
5. W. Charoenwet, P. Thongtanunam, V.-T. Pham, and C. Treude, "An empirical study of static analysis tools for secure code review," in Proc. 33rd ACM SIGSOFT Int. Symp. Softw. Testing Anal. (ISSTA), Vienna, Austria, Sep. 2024, pp. 691–703, doi: 10.1145/3650212.3680313.
6. B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?" in Proc. 35th Int. Conf. Softw. Eng. (ICSE), San Francisco, CA, USA, 2013, pp. 672–681.
7. C. Vassallo, S. Panichella, F. Palomba, S. Proksch, H. C. Gall, and A. Zaidman, "How developers engage with static analysis tools in different contexts," Empir. Softw. Eng., vol. 25, no. 2, pp. 1419–1457, 2020.
8. H.-M. Hu, Y. Liu, C. Chen, and X. Zhang, "An empirical study of suppressed static analysis warnings," Proc. ACM Softw. Eng., vol. 2, no. FSE, pp. 1–22, 2025.

9. P. Hegedűs and R. Ferenc, "Static code analysis alarms filtering reloaded: A new real-world dataset and its ML-based utilization," *IEEE Access*, vol. 10, pp. 55090–55101, 2022, doi: 10.1109/ACCESS.2022.3176865.
10. R. Yedida, R. Krishna, and T. Menzies, "How to find actionable static analysis warnings: A case study with FindBugs," *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, pp. 1830–1842, 2023.
11. H. J. Kang, K. L. Aw, and D. Lo, "Detecting false alarms from automatic static analysis tools: How far are we?" in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng. (ICSE)*, Pittsburgh, PA, USA, May 2022, pp. 698–709, doi: 10.1145/3510003.3510214.
12. W. Lee, W. Lee, D. Kang, K. Heo, H. Oh, and K. Yi, "Sound non-statistical clustering of static analysis alarms," *ACM Trans. Program. Lang. Syst.*, vol. 39, no. 4, Art. no. 16, Sep. 2017, doi: 10.1145/3095021.
13. N. Mansoor, T. Muske, A. Serebrenik, and B. Sharif, "An empirical assessment on merging and repositioning of static analysis alarms," in *Proc. IEEE 22nd Int. Work. Conf. Source Code Anal. Manip. (SCAM)*, Limassol, Cyprus, 2022, pp. 219–229, doi: 10.1109/SCAM55253.2022.00031.
14. L. Zhao, Y. Zhang, Y. Liu, Z. Chen, and X. Xia, "Software composition analysis for vulnerability detection: An empirical study on Java projects," in *Proc. 31st ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, San Francisco, CA, USA, Dec. 2023, pp. 960–972, doi: 10.1145/3611643.3616299.
15. R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: A systematic review," *Inf. Softw. Technol.*, vol. 141, Art. no. 106700, Jan. 2022, doi: 10.1016/j.infsof.2021.106700.
16. U. Koc, P. Saadatpanah, J. S. Foster, and A. A. Porter, "Learning a classifier for false positive error reports emitted by static code analysis tools," in *Proc. 1st ACM SIGPLAN Int. Workshop Mach. Learn. Program. Lang. (MAPL)*, Barcelona, Spain, 2017, pp. 35–42, doi: 10.1145/3088525.3088675.
17. S. S. Yerramreddy, A. Mordahl, U. Koc, S. Wei, J. S. Foster, M. Carpuat, and A. A. Porter, "An empirical assessment of machine learning approaches for triaging reports of static analysis tools," *Empir. Softw. Eng.*, vol. 28, no. 2, Art. no. 28, Mar. 2023, doi: 10.1007/s10664-022-10253-z.
18. T. Muske and A. Serebrenik, "Survey of approaches for postprocessing of static analysis alarms," *ACM Comput. Surv.*, vol. 55, no. 3, Art. no. 48, Mar. 2022, doi: 10.1145/3494521.
19. Z. P. Fry and W. Weimer, "Clustering static analysis defect reports to reduce maintenance costs," in *Proc. 20th IEEE Working Conf. Reverse Eng. (WCRE)*, Koblenz, Germany, 2013, pp. 341–350, doi: 10.1109/WCRE.2013.6671308.
20. Y. Pu, Z. Lin, X. Zhang, and L. Tan, "From noise to signal: Precisely identify affected packages of known vulnerabilities in npm ecosystem," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San San Diego, CA, USA, Feb. 2026, pp. 1–18.
21. F. Soltaniani, A. Mordahl, and S. Wei, "Learning to detect hardcoded secrets," *Empir. Softw. Eng.*, vol. 31, no. 1, Art. no. 12, Jan. 2026.
22. S. Chakraborty, R. Krishna, Y. Ding, and B. Ray, "Deep learning based vulnerability detection: Are we there yet?" *IEEE Trans. Softw. Eng.*, vol. 48, no. 9, pp. 3280–3296, Sep. 2022, doi: 10.1109/TSE.2021.3087402.
23. A. N. Angelopoulos and S. Bates, "Conformal prediction: A gentle introduction," *Found. Trends Mach. Learn.*, vol. 16, no. 4, pp. 494–591, 2023, doi: 10.1561/2200000101