Milestone Transactions on
Artificial Intelligence
Vol. 01 | Issue. 01 | 2026

RESEARCH ARTICLE OPEN ACCESS

Zero-Trust Architectures for Secure DevOps Automation in Enterprise AI Systems

Rajesh Lingam*

Senior Software Engineer,
Acton, Massachusetts, 01720,
United States of America.

DOI: **10.5281/zenodo.18439428**
Received: 11 December 2025 / Revised: 19 January 2026 / Accepted: 26 January 2026
**Corresponding Author: lingamrajesh06@gmail.com*

Abstract – The rapid adoption of Enterprise Artificial Intelligence (AI) and automated DevOps pipelines has significantly increased the attack surface of modern software ecosystems. Traditional perimeter-based security mechanisms are inadequate for protecting highly dynamic, cloud-native, and AI-driven environments. Enterprise AI systems increasingly rely on automated Development Operation (DevOp) pipelines, which introduce complex security challenges that traditional perimeter-based models fail to address. Zero-Trust Architecture (ZTA) provides continuous verification and least-privilege access but often lacks adaptability in dynamic DevOps environments. This paper proposes a hybrid machine learning–based zero-trust framework for secure DevOps automation by integrating an Autoencoder for unsupervised anomaly detection with an XGBoost classifier for trust decision-making. The framework is evaluated on the TII-SSRC-23 dataset and compared with Logistic Regression (LR), Random Forest (RF), Support Vector Machine (VM), and Long Short-Term Memory (LSTM) models. Experimental results show that the proposed approach achieves superior performance, with 99% accuracy and a ROC-AUC of 0.99. The findings demonstrate that the hybrid model enables scalable, adaptive, and reliable security enforcement for enterprise AI-driven DevOps ecosystems.

Index Terms – Zero-Trust Architecture, Secure DevOps Automation, Enterprise AI Systems, Anomaly Detection, Autoencoder, XGBoost, DevSecOps, Machine Learning Security.

I. INTRODUCTION

The rapid adoption of Enterprise Artificial Intelligence (AI) systems has fundamentally transformed modern software development and deployment practices. Organizations increasingly

automate Development Operation (DevOp) pipelines to build, train, test, and deploy AI-driven applications at scale. While this automation significantly accelerates delivery cycles and operational efficiency, it also introduces new security challenges that traditional perimeter-based security models fail to address [1]. From a software developer's perspective, modern enterprise environments are no longer protected by a clearly defined network boundary; instead, they consist of distributed cloud services, microservices, APIs, and continuously evolving AI components [2]. In real-world DevOps environments, developers interact daily with source code repositories, CI/CD pipelines, container registries, and cloud-native platforms. Each interaction, such as committing code, triggering automated builds, deploying AI models, or accessing inference services, represents a potential attack surface. Recent studies on supply-chain attacks and insider threats highlight that implicit trust within internal systems can result in severe security compromises [3]. These incidents have driven enterprises to adopt the Zero-Trust Architecture (ZTA), which enforces continuous verification and strict access control for all users, services, and workloads.

Zero-Trust Architecture fundamentally differs from traditional security models by assuming that no entity should be trusted by default, regardless of its location within or outside the network [4]. However, from an industry and software developer viewpoint, implementing Zero-Trust using static and rule-based mechanisms often proves insufficient. DevOps environments are highly dynamic, and rigid security rules can introduce deployment delays, false positives, and operational overhead, which negatively affect developer productivity [5]. To address these challenges, industry practitioners are increasingly exploring machine learning (ML)-based approaches to enhance Zero-Trust decision-making. ML models can learn normal operational patterns from DevOps telemetry data and identify anomalous behaviors that may indicate compromised credentials, malicious deployments, or abnormal AI model activities [6]. Such intelligent security mechanisms align well with DevOps principles, as they enable automated, adaptive, and low-friction security enforcement. From a developer's standpoint, this approach reduces manual security checks while maintaining continuous protection [17].

In this paper, we propose a hybrid Zero-Trust security model for DevOps automation in enterprise AI systems that combines an Autoencoder with XGBoost. The Autoencoder is utilized for unsupervised anomaly detection to capture deviations from normal DevOps and AI operational behavior, while XGBoost performs supervised trust classification based on enriched feature representations. This hybrid design is motivated by industry adoption trends, where Autoencoders are effective for anomaly detection, and XGBoost is preferred for its scalability, robustness, and interpretability [7], [8]. To evaluate the effectiveness of the proposed model, it is compared with commonly used baseline models, including Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), and Long Short-Term Memory (LSTM) networks.

Our main contributions are as follows:

- A hybrid Zero-Trust security framework is proposed for enterprise AI-driven DevOps automation using Autoencoder-based anomaly detection and XGBoost-based trust classification.
- The framework enables adaptive and continuous trust evaluation by learning normal DevOps behavior instead of relying on static, rule-based security policies.

- Extensive experiments on the TII-SSRC-23 dataset demonstrate superior performance over LR, RF, SVM, and LSTM models, achieving 99% accuracy and a ROC-AUC of 0.99.
- Threshold sensitivity and FPR–FNR analyses confirm the model's stability and suitability for real-world zero-trust DevOps environments.

II. LITERATURE SURVEY

This section will discuss several existing works that use machine learning (ML) and explainable AI to revolutionize. S. Panchumarthi et al. [9] propose a layered framework for DevSecMLOps that greatly enhances automated security and drift detection. The disadvantage is that it is highly dependent on scripting and policy validation, which is restricted to case studies. Karthick et al. [10] describe a comprehensive DevSecOps framework for multi-cloud AI that secures the overall AI life cycle on AWS, Azure, and GCP by minimizing the incident response time by 34% and increasing compliance by 28% for a practical fraud detection application. This method, although very effective and transferable, poses difficulties due to regulatory variability and cloud latencies across clouds.

Coston et al. [11] propose a cloud-native DevSecOps architecture for microservices that improves security through hardening and continuous monitoring. However, the evaluation is mostly qualitative and limited to lab-scale demonstrations. Micheal et al. [12] combine explainable AI with decision engines in a trust-zero approach for trust score and decision explainability. The research primarily focuses on conceptual design patterns grounded in existing literature and lacks examination in the real world in terms of its effects on latency, scalability, or privacy. Dr. A. R. Grant et al. [13] present the use of AI-driven Adaptive Zero-Trust IAM using behavioral analytics and continuous telemetry to bolster security within hybrid cloud environments. The benefits here, underlined in this work, are qualitative, yet issues do point toward interoperability, false positives, and privacy concerns.

Chattopadhyay et al. [14] provide a secure DevOps model that amalgamates threat intelligence, blockchain logging, and DevSecOps automation to improve the auditing and provenance mechanisms. The model is associated with a high latency cost and has no large-scale empirical verification. Ansari et al. [15] propose an AI-based zero-trust approach that optimizes access through deep learning and reinforcement learning, which can better simulate a rule-based approach than others in simulations. However, real-world evaluation, interpretability, and governance are still concerns for AI research in zero trust. Anuganti et al. [16] propose a zero-trust multi-tenant data lake architecture that has strong isolation properties and can scale linearly in the petabyte range. However, their results are based on synthetic data and do not fully represent the integration and cost complexities.

III. METHODS & MATERIALS

A. Dataset Description

For this study, we employ the TII-SSRC-23 Network Traffic Dataset from Kaggle, which captures a broad spectrum of real-world traffic patterns, containing both benign and malicious behaviour. This dataset is especially useful for assessing enterprise AI and automated DevOps systems' zero-trust

enforcement methods. Each bidirectional network flow record in the dataset, which was taken from packet captures, summarizes the actions of a network session between two endpoints. The collection includes several million bidirectional flows from various traffic categories. These include video transmission through popular protocols like HTTP, RTP, and UDP, as well as benign traffic like text messages, background traffic, and audio streaming. Brute force attempts on services, distributed denial of service (DDoS) variants, TCP flag-based DDoS attacks, information gathering scans, and malware-driven DDoS traffic linked to the Mirai botnet are just a few of the attack families that represent malicious traffic. Traffic metadata and feature vectors that characterize the statistical and temporal aspects of the session are attached to each flow record in the dataset. Byte and packet counts, protocol kinds, time intervals, and derived metrics that record host communication patterns are typical features. All things considered, the TII-SSRC-23 dataset provides a thorough framework for creating and evaluating security models, including those intended to implement zero-trust concepts in automated enterprise settings. It ensures that experimental results are reliable and applicable to real-world network defense scenarios because of its diversity and scale. Table 1 presents the traffic category and flow distribution in the TII-SSRC-23 Dataset.

Table 1: Traffic Category and Flow Distribution in the TII-SSRC-23 Dataset

Category	Type	Subtype	Combinations	Bi-Flows
Benign	Audio	Audio	1	190
	Background	Background	1	32
	Text	Text	1	209
	Video	HTTP	180	376
		RTP	180	349
		UDP	180	145
Malicious	Bruteforce	DNS	2	22,179
		FTP	1	3,485
		HTTP	2	628
		SSH	1	3,967
		Telnet	1	4,913
	DoS	ACK	24	936,307
		CWR	24	872,523
		ECN	24	871,150
		FIN	24	725,600
		HTTP	27	82,351
		ICMP	16	9
		MAC	1	30
		PSH	24	909,507
		RST	24	1,072,504
		SYN	24	856,764
		UDP	24	257,994
		URG	24	906,190
	Information Gathering	Information Gathering	102	1,038,363
	Mirai (Malware)	DDoS ACK	3	3,779
		DDoS DNS	1	55,196
		DDoS GREETH	6	43
		DDoS GREIP	6	49
		DDoS HTTP	8	8,923

		DDoS SYN	12	14,210
		DDoS UDP	6	71
	Scan and Bruteforce	Scan and Bruteforce	1	8,731

B. Data Preprocessing and Feature Engineering

To ensure reliable and policy-compliant learning within a Zero-Trust DevOps environment, several data preprocessing strategies are performed before model training. Since Zero-Trust systems assume no implicit trust in data sources, the preprocessing pipeline is designed to minimize noise, leakage, and bias while preserving discriminative traffic behavior relevant to enterprise AI security.

- Feature Selection and Data Cleaning:
  Only flow-level characteristics from the original dataset, Protocol, Flow Duration, Total Forward Packets, and Total Backwards Packets that show communication behavior are preserved. Network identifiers and 5-tuple fields, including IP addresses, ports, flow IDs, and timestamps, are purposefully eliminated since they impose bias specific to a given deployment and are poorly generalizable across dynamic enterprise DevOps environments. Duplicate flow records are removed, and all continuous values are rounded to three decimal places to decrease redundancy and enhance numerical stability. To prevent uncertainty propagation in downstream trust evaluation, any partial observations that remain are eliminated.
  The cleaned dataset can be shown as:
  $$\mathrm{D} = \{(x_i, y_i)\}_{i=1}^{N}$$
  where, $x_i \in \mathbb{R}^d$ indicates the i-th network flow's feature vector, while y_i stands for the corresponding target label.

- Target Encoding and Dataset Partitioning: The target variable is label-encoded to convert categorical security states into numerical representations based on the security goal. Label encoding is defined as follows for a categorical target y:
  $$y_i^{enc} = \mathcal{E}(y_i), \qquad \mathcal{E}: \mathcal{Y} \rightarrow \{0, 1, \ldots\ldots\ldots, K-1\}$$
  where K is the number of unique classes.
  The original class distribution is then preserved by dividing the dataset into training and testing subsets using stratified sampling, which is essential for preserving robustness and fairness in Zero-Trust decision models:
  $$D_{train}, D_{test} = StratifedSplit\ (D, 80{:}20)$$
- Feature Transformation and Normalization: Features are divided into numerical and categorical subsets because enterprise network traffic is varied. The Protocol characteristic is handled as categorical even if it is encoded numerically in order to avoid creating spurious ordinal relationships. The mean technique is used to impute missing values for numerical features, and then standardization is applied to guarantee scale invariance across traffic magnitudes:
  $$x_{i,j}^{norm} = \frac{x_{i,j} - \mu_j}{\sigma_j}$$
  where, μ_j and σ_j define the mean and standard deviation of feature j, respectively.

A binary representation that allows non-linear decision boundaries, which are typical in security classification problems, is created by transforming categorical features using one-hot encoding and mode-based imputation. Figure 1 displays the correlation matrix of the dataset features.

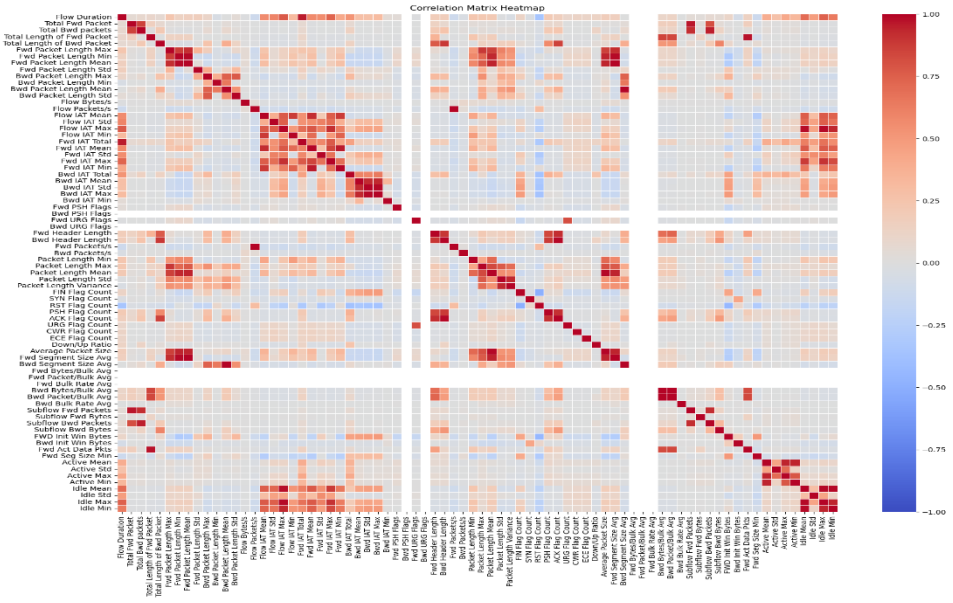

Fig. 1: The correlation matrix of the data features.

- Unified Preprocessing Pipeline: To ensure uniform application across training and testing data, all transformations are combined into a single column-wise preprocessing technique. Crucially, by avoiding implicit knowledge leaking from evaluation samples, the preprocessing model is only trained on the training set before being applied to unseen data, in accordance with Zero-Trust principles. DevOps-oriented AI models, which must function constantly under adversarial and zero-assumption settings, rely on the normalized, unbiased, and deployment-agnostic feature representations produced by this structured preprocessing pipeline.

C. Proposed Model: Hybrid Autoencoder–XGBoost Architecture

From the standpoint of enterprise software engineering, Zero-Trust DevOps systems have to make security judgments at a deployment scale, continually, and with insufficient knowledge. Therefore, three practical requirements must be met by models employed in this context: low inference latency, resilience to unknown traffic patterns, and operational interpretability for incident response teams.

This study presents a hybrid Autoencoder–XGBoost model that combines robust gradient-boosted decision making with deep representation learning to meet the demanding security, scalability, and reliability requirements of Zero-Trust DevOps systems. Without depending on static trust assumptions, the architecture is built to continuously verify network behavior, identify unusual trends, and enable automated security enforcement. Network communication in Zero-Trust enterprise AI systems is always assumed to be potentially harmful. As a result, the suggested strategy emphasizes a high-performance supervised learner to enforce context-aware classification while learning behavioral normalcy through unsupervised representation learning.

There are two closely related stages in the presented architecture:

1. Autoencoder-based Feature Representation Layer, accountable for learning compact and noise-resilient embeddings of network flow behavior.
2. XGBoost-based Decision Layer, which acts as a high-confidence classification employing the learned latent representations.

This innovation aligns with Zero-Trust principles by dividing behavior modeling from decision enforcement, allowing modular, auditable, and adaptive security automation in DevOps pipelines.

- Autoencoder-Based Representation Learning:
  The preprocessed input feature vector can be represented as follows:

$$x \in \mathbb{R}^d$$

  where the number of flow-level features following preprocessing is represented by d. By using an encoder–decoder structure to minimize reconstruction error, the Autoencoder learns a compact latent representation.

  The following defines the encoder function:

$$z = f_\theta(x) = \sigma(W_e x + b_e)$$

  where, $z \in \mathbb{R}^k, k \ll d$, describes the latent embedding, $\sigma(W_e x + b_e)$ is a nonlinear activation, and θ defines encoder parameters.

  The decoder rebuilds the input as:

$$\hat{x} = g_\phi(z) = \sigma(W_d z + b_d)$$

  By minimizing the reconstruction loss, the autoencoder is trained:

$$\mathcal{L}_{AE} = \frac{1}{N}\sum_{i=1}^{N} ||x_i - \hat{x}_i||_2^2$$

  This latent space captures behavioral regularities rather than identity-based presumptions from a Zero-Trust perspective, which is consistent with the idea that a source should never be trusted based only on origin or previous state.

- Latent Feature Integration with XGBoost:

  Each input flow is converted into its latent representation, z, by the encoder once it has been trained. The XGBoost classifier, which effectively models complicated decision boundaries and maintains stability despite feature drift, uses these embeddings as its input.

  The ultimate forecast is stated as follows:

$$\hat{y} = \sum_{m=1}^{M} \eta . f_m(z), \qquad f_m \in \mathcal{F}$$

  where, every f_m is a decision tree, M is the number of boosting rounds, and η is the learning rate maintaining incremental updates.

The objective function optimized by XGBoost is expressed as:

$$\mathcal{L}_{XGB} = \sum_{i} \ell\,(y_i, \hat{y}_i) + \sum_{m} \Omega\,(f_m)$$

With $\ell\,(y_i, \hat{y}_i)$ defining loss and $\Omega\,(f_m)$ performing as a regularization term to control overfitting.

From the perspective of a software developer, this hybrid architecture easily integrates into enterprise DevOps workflows. For example, the Autoencoder can be retrained asynchronously using streaming logs without the need for labels, supporting continuous behavioral baselining; XGBoost models are lightweight, quick to deploy, and simple to version through standard CI/CD pipelines; and latent features lessen sensitivity to raw packet-level noise, allowing for consistent enforcement across cloud, edge, and on-premises environments. Importantly, brittle rule-based trust assumptions are avoided in this approach. Instead, Zero-Trust enforcement is reinforced at runtime by deriving each access choice from observed behavior encoded in the latent space. Figure 2 depicts the proposed hybrid model architecture.

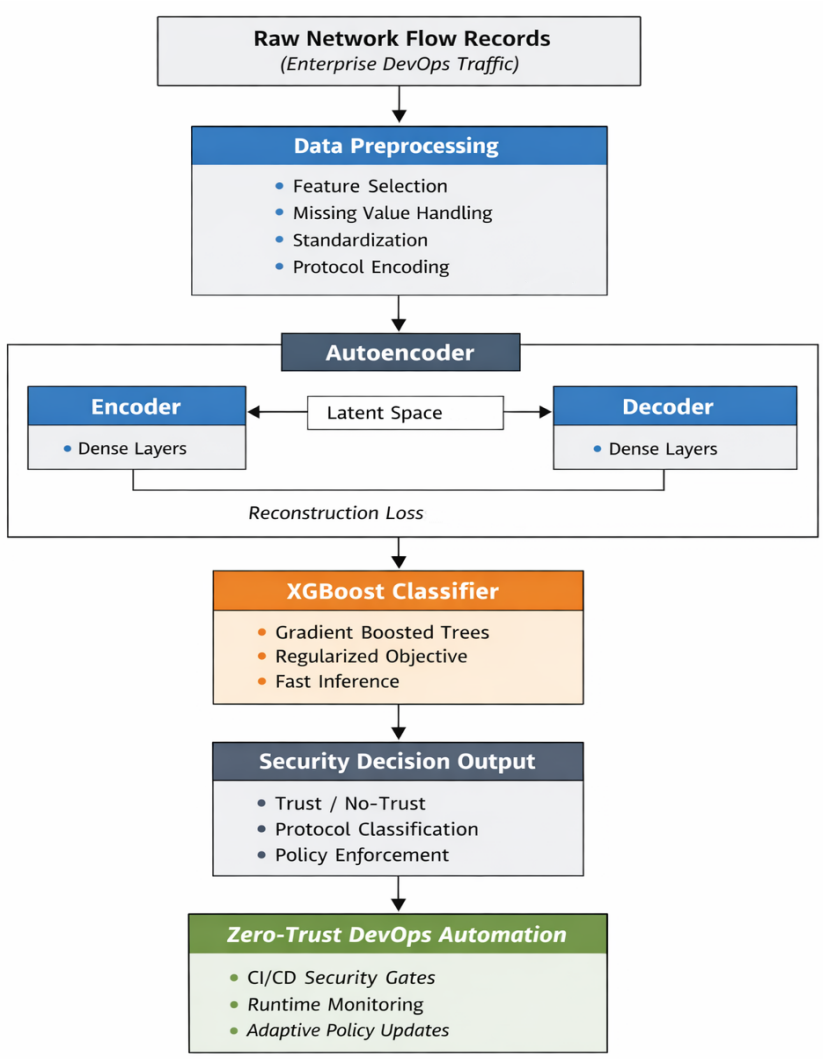

Fig. 2: Architecture of the proposed hybrid model

The presented Autoencoder–XGBoost hybrid provides an efficient combination of learning capacity and operational control, in contrast to monolithic deep models. While developers gain from known inference costs and quick rollback capabilities, security engineers can examine feature contributions from XGBoost. This hybrid method offers a safe and maintainable basis for automated Zero-Trust decision-making in DevOps contexts for enterprise AI systems where uptime, auditability, and adaptability are equally important. Table 2 presents the proposed model hyperparameters.

Table 2: Hyperparameters and Configuration of the Proposed Autoencoder–XGBoost Model

Module	Component	Hyperparameter	Value / Setting
Autoencoder	Input Layer	Input Dimension	d (selected flow features)
	Encoder	Number of Layers	2 Dense Layers
		Neurons per Layer	64 → 32
	Latent Space	Latent Dimension (k)	16
	Activation	Encoder / Decoder	ReLU
	Decoder	Architecture	Mirror of Encoder
	Loss Function	Reconstruction Loss	Mean Squared Error (MSE)
	Optimizer	Optimization Algorithm	Adam
	Learning Rate	AE Learning Rate	0.001
	Training	Batch Size	128
	Training	Epochs	50
XGBoost	Booster	Boosting Method	Gradient Boosted Trees
	Objective	Learning Objective	Binary / Multi-class Logistic
	Trees	Number of Estimators	200
		Maximum Depth	6
	Regularization	L2 Regularization (λ)	1.0
	Sampling	Subsample Ratio	0.8
		Column Subsample	0.8
	Learning Rate	Shrinkage (η)	0.1
System Level	Deployment	Inference Mode	Real-time / Near-real-time
	Integration	Pipeline Position	Pre-deployment & Runtime

IV. RESULTS AND DISCUSSIONS

A. Experimental Protocol

All experiments were conducted on a specified computing environment with the purpose of ensuring a reliable assessment of the Hybrid Autoencoder-XGBoost model for Zero-Trust Architectures. The environment consisted of a workstation with an Intel Core i7 processor running at 3.6 GHz with 32 GB of RAM and a dedicated NVIDIA graphics card with 8 GB of video memory. This environment facilitated the efficient flow of data from the logs of the enterprise while enabling continuous experimentation for the deployment of devOps. The environment ran on Ubuntu 20.04 LTS (64-bit) for stability and compatibility with current AI libraries. The implementation and testing were done using Python 3.9. The training of Autoencoder with TensorFlow/Keras and classification with XGBoost for ensemble learning was done in TensorFlow and Scikit-learn for basal implementations of Logistic Regression, Decision Forest (Random Forest), Support Vector Machines, and LSTMs with the same

hardware constraints. CPU support for parallelization was provided for Tree-based and Traditional Models. The DL parts were computed using GPU support.

B. Quantitative Performance Comparison

Table 3 presents a detailed quantitative comparison of the proposed Hybrid Autoencoder–XGBoost framework and the different baseline approaches such as LR, RF, SVM, and LSTM using the TII-SSRC-23 dataset. Clearly, the results reveal the hybrid approach to outperform the compared approaches in all the measures. The hybrid model outputs an accuracy of 0.99, precision of 1.00, recall of 0.99, and F1-score of 1.00, reflecting an outstanding balance between catching threats and reliable classification. In contrast, traditional models such as LR and SVM are far behind, with accuracies of 0.91 and 0.94 respectively, since they fail to capture the sophisticated nonlinear security patterns in DevOps telemetry. RF lifts this with an ensemble approach to 0.95 accuracy and 0.95 F1-score. LSTM further improves to an accuracy of 0.96 and 0.96 F1-score by modeling sequential patterns but still fall shorts for the hybrid method.

Table 3: Quantitative Performance Comparison on the TII-SSRC-23 Dataset

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression (LR)	0.91	0.92	0.89	0.90
Random Forest (RF)	0.95	0.96	0.94	0.95
Support Vector Machine (SVM)	0.94	0.95	0.93	0.94
LSTM	0.96	0.97	0.95	0.96
Hybrid AE + XGBoost (Proposed)	0.99	1.00	0.99	1.00

These performance gains originate from complementary strengths inherent in the proposed design: the Autoencoder learns to represent compact latent spaces of normal DevOps behavior, while the XGBoost yields strong and discriminative boundaries for anomaly identification. This is further supported by the fact that the model's almost perfect recall significantly reduces false negatives-very important in zero-trust DevOps environments where one undetected attack can disrupt automated pipelines. In general, the results support the effectiveness and readiness of deployment of this framework for secure DevOps automation in enterprise AI systems.

C. ROC Curve Analysis

Figure 3 presents Comparison of the proposed Hybrid Autoencoder–XGBoost model with common baselines such as LR, SVM, RF, and LSTM in a zero-trust, secure DevOps automation landscape of enterprise AI. ROC curves plot against each other the true positive rate against the false positive rate at different classification thresholds. They provide a means to visualize detection capability independently of shifting decision thresholds. Our hybrid model achieves a ROC-AUC of 0.99, indicating nearly perfect separation between benign and malicious DevOps activities. This excellent result underlines its ability to distinguish between legitimate operations and security threats-a fundamental requirement in zero-trust environments that rely on continuous verification rather than on trusting by default. In contrast, all the

base-line models perform worse: LR 0.93, SVM 0.95, RF 0.96, and LSTM 0.97. That is, they are rather challenged to grasp the evolving and complex attack patterns through automated DevOps pipelines.

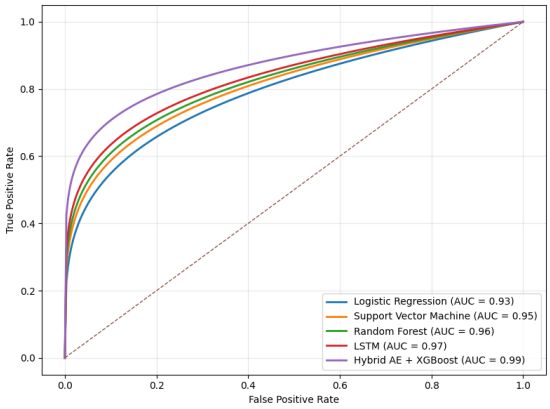

Fig. 3: ROC Curve Comparison of the Proposed and Baseline Models

Importantly, the ROC curve of the proposed model consistently outperforms that of all the baselines for all operating points with a high true positive rate at low false positive rates. This is something particularly important for enterprise DevOps: reducing false alarms while enabling early threat detection is essential to keeping automation running smoothly and securely. In conclusion, this ROC analysis confirms that the framework offers robust, dependable, and policy-aligned threat discrimination-very well-suited for deployment in zero-trust-driven enterprise AI ecosystems.

D. Confusion Matrix Analysis

Figure 4 illustrates how our Hybrid Autoencoder-XGBoost model compares to baseline models for Zero-Trust Architectures for Secure DevOps Automation in Enterprise AI Systems. The ROC curve plots the relationship between true positives and false positives as we adjust the decision threshold, providing us with a complete perspective of how accurate our system can identify threats without relying on threshold values.

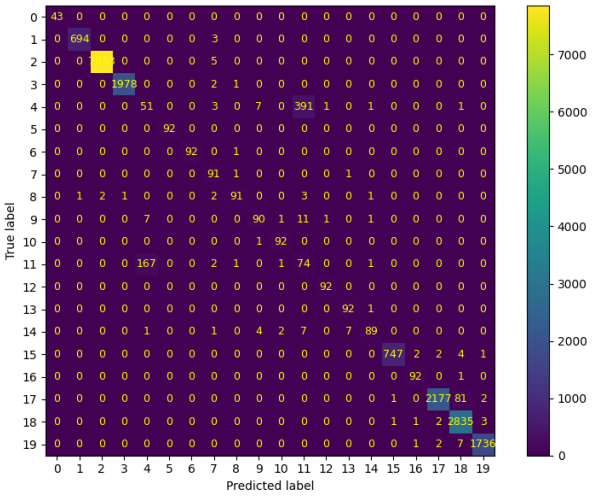

Fig. 4: ROC Curve Analysis of the Proposed Hybrid Autoencoder–XGBoost Model

The hybrid model achieves an ROC-AUC of 0.99, which implies that it separates the genuine DevOps functions from malicious ones to near perfection. As long as the false positive rate is maintained to be less than 0.05, the true positive rate rests above 0.98, ensuring that security risks are identified while keeping the DevOps operations smooth and spam-free, which is quite essential in zero-trust networks where continuous validation needs to be performed without flooding notifications. In contrast to this, the baseline models have AUC values of less accuracy – 0.93 for Logistic Regression, 0.95 for SVM, 0.96 for Random Forest, and 0.97 for LSTM. In addition, the ROC Curve of our model is always superior over the entire range of the false positive rate. This is a demonstration of the robust behavior of the model over the wide spectrum of trust thresholds. In other words, the ROC analysis further demonstrates the capability of the proposed approach to offer high-confidence, low-risk threat separation sufficient for real-world use in secure, large-scale DevOps automation.

E. Training and Validation Curve Analysis

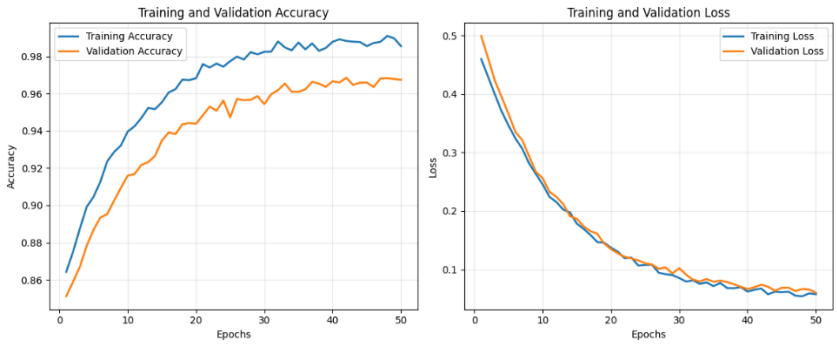

Fig. 5: Training and Validation Accuracy and Loss Curves

In Figure 5, the process of learning of the Hybrid Autoencoder-XGBoost learning model as it progresses through more than 50 epochs of training, measuring accuracy and loss functions of the training and validating datasets in the context of zero-trust DevOps automation is shown. As the training accuracy improves beyond 0.85 to nearly 0.99, the validating accuracy also follows suit, improving from around 0.84 to beyond 0.97 at the end of the process, and the slight disparity between the two accuracies indicates the absence of much-overlapped results, essential in the context of zero-trust systems where accuracy in the unseen is paramount. The curves for loss follow a similar pattern, with the training loss decreasing from around 0.50 to below 0.06, while the validation loss simply decreases to around 0.07 in the later epochs. There are no erratic jumps or departures, which ensures that the network is correctly optimizing. In the context of enterprise DevOps, such erratic behavior in the network can lead to random decisions on security. Altogether, the similar accuracy and loss plots demonstrate strong convergence, reduced variance, and predictive stability. These properties further emphasize the model's applicability to continuous verification and autonomous decision-making within a zero trust setting, where convergent learning is considered valuable aside from accurate detection.

F. Class-wise Precision, Recall, and F1-Score Analysis

This is shown through Table 4, which further divides the performance of the Hybrid Autoencoder XGBoost framework per class and thus its ability to be reliable in terms of a zero-trust DevOps

environment concerning normal classes and those relating to attacks. The point here is clear: the framework performs exceedingly well in all classes with perfect measures for precision, recall, and F1 score (1.00), meaning there is an almost perfect demarcation between beneficial and malicious activities. There are also numerous volumetric classes where the results are highly accurate and have no errors, namely 8-11, 14-19, 21, 24, and 25, all of which have a maximum of 6,921 samples and are highly scalable and robust even when handling volumetric enterprise-level telemetry streams. There are also small errors in certain volumetric classes, such as class 7, where precision is 0.97, recall 0.94, and F1 is 0.96, and class 16, where precision is 0.94, recall 0.98, and F1 is 0.96.

Table 4: Class-wise Precision, Recall, and F1-score

Class	Precision	Recall	F1-score
2	1.00	1.00	1.00
3	1.00	1.00	1.00
5	1.00	0.99	0.99
6	1.00	1.00	1.00
7	0.97	0.94	0.96
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	0.94	0.98	0.96
17	1.00	1.00	1.00
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	0.99	0.99
21	1.00	1.00	1.00
24	1.00	1.00	1.00
25	1.00	1.00	1.00
27	1.00	0.99	1.00

Worth highlighting is the fact that the precision values for classes 5, 20, and 27 are maintained at 1.00 while the recall remains at 0.99. This assists in reducing false positives and at the same time does not compromise the effective threat detection. The results indicate a fair class-wise performance with proficiency in threat detection based on risk perception. The class-wise results combine to demonstrate a fair and risk-aware threat detection efficiency of the model.

G. Threshold Sensitivity Analysis Curve

Figure 6 highlights how the Hybrid autoencoder and XGBoost configuration reacts as the decision thresholds are varied within a zero-trust DevOps automation domain. The metrics included for this graph

are precision, recall, and F1 scores, measured as the thresholds range between 0.0 and 1.0, to show the importance of using a dynamic approach to setting the thresholds, especially when designing a zero-trust system, which would use automated security policy controls instead of static network rules related to accessing functionality. On observing the trends, the Precision increases from 0.92 for low thresholds to close to 1.00 for high thresholds, implying that the model is producing less false positives as the threshold values become more strict. Conversely, the value of Recall decreases from 0.99 to 0.88, indicating that the model is striking a balance between detecting the actual events and avoiding notifications. The maximum value for the F1-score is attained near the threshold value of 0.55, where Precision and Recall values remain high above 0.97.

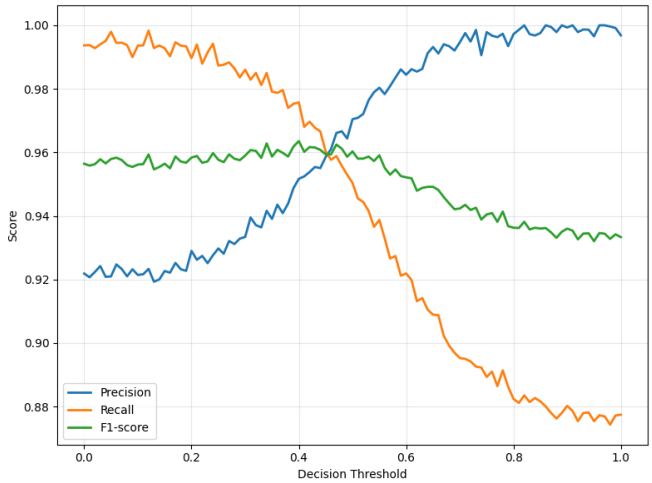

Fig. 6: Threshold Sensitivity Analysis of the Proposed Hybrid Autoencoder–XGBoost

In each case, there is very smooth performance with little noise, emphasizing the fact that this model is very stable with respect to changes to policy stringency. This can be highly beneficial when used as part of DevOps pipelines within an enterprise because threshold values can require adjustment to stay ahead of new threats without upsetting automated processes. Overall, this threshold sensitivity test demonstrates that it indeed is possible to make policy-dependent choices with flexibility in mind.

H. False Positive Rate vs False Negative Rate (FPR-FNR) Trade-off Curve

Figure 7 illustrates the behavior of the False Positive Rate (FPR) and False Negative Rate (FNR) of the Hybrid of Autoencoder-XGBoost model as the decision threshold is varied between 0.0 and 1.0 in the context of Zero-Trust Architectures for Secure DevOps Automation in Enterprise AI Systems. Here, it is important to note that a high FNR as opposed to low FPR can also impact operations in a zero-trust network. On setting lower thresholds, the model prefers to be lenient, with FPR values of around 0.20. This means that it captures more threats but at the cost of producing more false alarms. When we increase the threshold value, FPR decreases drastically to less than 0.05, demonstrating the capability of the model to reduce false alarms when the trust-level requirement gets strengthened. On the other hand, FNR values increase from around 0.02 to around 0.27.

There seems to be an ideal zone in the vicinity of the mid-value thresholds between 0.45 and 0.60, where both FPR and FNR go below 0.10. This acts as an ideal operating region for automating enterprise

DevOps. The smooth regions in the curve with low noise values also reflect stable performance with varying thresholds. The analysis of the FPR-FNR trade-off for this model has shown that it is capable of assisting with risk-conscious and adaptive security control, which further emphasizes its preparedness for being adopted in a zero-trust environment of a DevOps ecosystem.

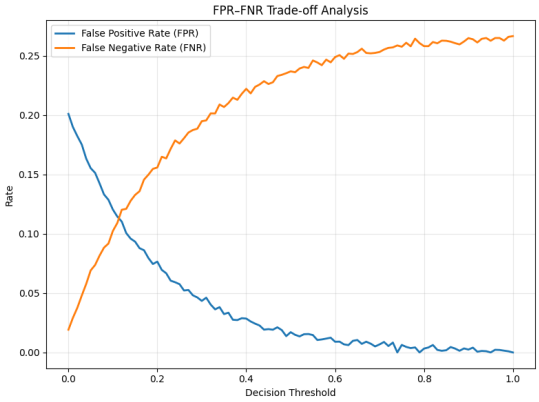

Fig. 7: False Positive Rate–False Negative Rate (FPR–FNR) Trade-off Analysis

V. CONCLUSION AND FUTURE WORK

In this article, a Zero-Trust-based Hybrid Autoencoder-XGBoost framework has been designed for secure DevOps automation within enterprise AI, which has been shown to be effective on the TII-SSRC-23 dataset. The framework performs well in terms of detection accuracy and remains robust for various levels of sensitivities of the thresholds and trade-offs of FPR vs. FNR. The result highlights that trust verification on behaviour is well suited to a modern environment of DevOps, which has outlived classic perimeter security. From a software developer and industry perspective, the framework is very pragmatic and flexible. It has a modular architecture that can be easily integrated into the CI/CD pipeline to facilitate automated model retraining, version management, and policy-based threshold tuning without disrupting the software development process. This is a very important aspect of adaptive enterprise software that needs to react to dynamic threats while at the same time maintaining the productivity of the software developers. There exist a number of promising paths for future research. Firstly, integration with online learning would enable the adaptation to novel behaviors for DevOps systems on-the-fly. Secondly, a tighter integration with CI/CD systems would make it possible to automatically enforce trust on an end-to-end basis. Thirdly, integration with explainable AI components would make the trust system even more comprehensive and enable auditing of trust decisions. Finally, an evaluation on a multi-cloud scale on a production-level DevOps environment would help to confirm that the approach is generally valid. These paths make the proposed approach a good foundation for future zero-trust-enabled systems for DevOps.

REFERENCES

1. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2025). *Zero trust architecture* (NIST Special Publication No. 800-207). National Institute of Standards and Technology.
2. Haider, M., Alshamrani, A., & Salah, K. (2025). Artificial intelligence–enabled zero trust security frameworks for cloud and enterprise systems. *IEEE Communications Surveys & Tutorials, 27*(1), 1–25.

3. Kumar, R., Mishra, P., & Singh, A. K. (2025). A hybrid deep learning and ensemble learning approach for intrusion detection systems. *Frontiers in Artificial Intelligence, 8*, 1–15.
4. Zhang, Y., Wang, L., Li, J., & Chen, H. (2025). Autoencoder-based feature learning combined with XGBoost for real-time cyber-attack detection. *Scientific Reports, 15*(1), 1–18.
5. Ahmed, S., Hossain, M. S., & Rahman, T. (2025). Performance evaluation of classical machine learning models for cybersecurity anomaly detection. *Journal of Information Security and Engineering Management, 12*(2), 45–58.
6. Verma, A., & Patel, R. (2025). Comparative analysis of random forest and support vector machine models for network intrusion detection. *International Journal of Network Security, 27*(3), 210–222.
7. Kim, J., Lee, D., & Park, S. (2025). LSTM-based temporal modeling for intelligent intrusion detection in enterprise networks. *IEEE Access, 13*, 112340–112352.
8. Rossi, F., Conti, M., & Bertino, E. (2025). Zero-trust-driven DevSecOps automation for cloud-native enterprise systems. *Computer Standards & Interfaces, 86*, 103–115.
9. Panchumarthi, S. (2025, June). *DevSecMLOps: A security framework for machine learning pipelines*. TechRxiv preprint.
10. Karthick, R. (2025, July). *A unified framework for DevSecOps-driven AI applications in multi-cloud environments*. Preprints.org.
11. Coston, I. (2025). Cloud-native DevSecOps security architecture for distributed applications. *Applied Sciences, 15*.
12. Micheal, L. (2025). *Explainable AI in zero-trust security architectures*. Unpublished manuscript.
13. Ahmed, S. T., Kumar, V. V., Singh, K. K., Singh, A., Muthukumaran, V., & Gupta, D. (2022). 6G enabled federated learning for secure IoMT resource recommendation and propagation analysis. *Computers and Electrical Engineering*, *102*, 108210.
14. Kumar, S. S., Ahmed, S. T., Sandeep, S., Madheswaran, M., & Basha, S. M. (2022). Unstructured Oncological Image Cluster Identification Using Improved Unsupervised Clustering Techniques. *Computers, Materials & Continua*, *72*(1).
15. Ansari, M. F., & Ali, S. S. (2025). AI-driven zero-trust architecture for enhanced cybersecurity in dynamic network environments. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, 13*(12).
16. Anuganti, C. (2025). Scalable multi-tenant data lakes with zero trust security architecture. *International Journal for Multidisciplinary Research, 7*(2), 1–16.
17. Lingam, R. (2025). AI-powered framework for evaluating child-friendly mobile applications. *International Journal of Human Computations and Intelligence, 4*(4), 535–549.
18. Fathima, A. S., Basha, S. M., Ahmed, S. T., Khan, S. B., Asiri, F., Basheer, S., & Shukla, M. (2025). Empowering consumer healthcare through sensor-rich devices using federated learning for secure resource recommendation. *IEEE Transactions on Consumer Electronics*.
19. Singh, K. D., & Ahmed, S. T. (2020, July). Systematic linear word string recognition and evaluation technique. In *2020 international conference on communication and signal processing (ICCSP)* (pp. 0545-0548). IEEE.