National College of Ireland

Scalable Threat Assessment Machine Learning Framework for CI/CD DevSecOps Pipelines

MSc Research Project
Cloud Computing

Akhila Kandadi
Student ID: 23386347

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Akhila Kandadi
Student ID:	23386347
Programme:	Cloud Computing
Year:	2018
Module:	MSc Research Project
Supervisor:	Shaguna Gupta
Submission Due Date:	20/12/2018
Project Title:	Scalable Threat Assessment Machine Learning Framework for CI/CD DevSecOps Pipelines
Word Count:	XXX
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th November 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Scalable Threat Assessment Machine Learning Framework for CI/CD DevSecOps Pipelines

Akhila Kandadi
23386347

Abstract

The current CI/CD pipelines using machine learning (ML) models to predict, detect, and respond to security risks are still, however, vulnerable to smart cybersecurity attacks that challenge the traditional tools used for security. These approaches to policing security systems create a hurdle to the deployment of code since they face the problem of overhead on developers caused by frequent false positive incidences. The security function is an essential part of the product in the DevSecOps culture. Moreover, enabling teams to be fully automated with the rapid growth of technology is a challenge. The hybrid transformer-Graph Neural Network (tGNN) ensemble proposed in this research is a groundbreaking model, focusing on the combination of the temporal pattern recognition offered by transformer networks with the structural relationship analysis of GNN that allows the detection of attacks in real-time. In the proposed framework, the tGNN can detect multi-stage attacks in real-time due to the fusion of transformers and GNNs. It can be trained using the CICIDS-2017 dataset, augmented with synthetic data generation by Conditional Tabular Generative Adversarial Network (CTGAN). It is designed to be implemented in a production-ready containerized environment through Docker/Kubernetes. In this research, we envisage that the hybrid tGNN ensemble will have a better accuracy, a lower false-positive rate, and a higher event throughput power than the existing methods that do not have the temporal-spatial analysis ability. The framework bridges the critical integration gap between advanced ML capabilities and practical DevSecOps requirements.

1 Introduction

1.1 Background and Motivation

The changes caused by cybersecurity threats have triggered a shift to the widespread adoption of DevSecOps as a security paradigm in the cloud era, where, on average, organizations experience 1,100 cyberattacks per week, an increase of 38% from previous weekly averages (Fu et al., 2025). The security of CI/CD pipelines, which see an average of 1000 daily code commits made within a shortened time across multiple microservices, is being cracked with current threats more than ever. The journey from being reactive to now being a proactive enterprise with intelligent threat detectors is a testament to the transformative power of the coupled DevOps practices with machine learning. Transformer architectures' success in deep learning (DL), especially those originally designed for natural language processing (NLP), has shown particular success in reducing the error

rates in predicting the sequence of security incidents, with some achieving levels below 3% (Ullah et al., 2023). GNNs are an advantageous complement by capturing the structure of the system as a graph and enabling multi-dimensional threat assessment with both temporal and spatial aspects of security data (Zhong et al., 2024). The deployment of these technologies in DevSecOps frameworks not only solves the primary challenge of many organizations today, i.e., meeting customers' demands for rapid deliveries with the best security practices in an increasingly dangerous digital environment but also furthers the digital transformation. The work presented here builds upon the work of Alserhani and Aljared (2023), who achieved 99% accuracy using ensemble methods on the UNSW-NB15 dataset. We go a step further by integrating the pipeline's security in real-time, while the state-of-the-art hybrid model is working on the data simultaneously. Whoever will benefit more from this research are DevSecOps teams with the need for automatic security solutions, companies that follow the shift-left security approach, and the broader cybersecurity industry that, as the result of this project, will advance the ML-based threat detection methodology. The present paper creates a tangible benefit for the production process, which is the formalization of sophisticated security systems in the deployment of new generation threats in a seamless manner. This will in turn lead to the preservation of the operational efficiency, which is critical for the business to have a competitive edge in digital.

1.2 Problem Statement

Current CI/CD DevSecOps environments have exposed security tools powered by ML models to critical issues as they run mostly without connection to development workflows, and false-negative rates persist at 40%, thus causing alert storming and operational overhead (Fu et al., 2025). A good proportion of the self-learning capability of development pipelines with dynamic code threats that many approaches cannot capture remains unexploited by the batch-process-only mentality of the original paper. By their nature, traditional ensemble methods are not able to derive the temporal dependencies and structural relationships that are complex through modern distributed systems, thus resulting in null detections for orchestrated attacks. The present research is addressing these problems through a novel hybrid tGNN ensemble architecture that processes multimodal security data from CI/CD pipelines, container registries, and system logs in real-time. It will learn and adapt continuously with automated retraining being conducted by use of concept drift detection, thus ensuring that the models are always reaching their highest relevance as the threats evolve. The native GitLab CI/CD integration through webhooks enables serendipitous security assessment within the existing workflows, while the CT-GAN synthetic data generation is more effective in dealing with class imbalance than the conventional SMOTE techniques employed (Alabdulwahab et al., 2023). In case this proves successful, then the work will affirm that the hybrid neural topologies can achieve smaller proportions of false positives and high-performance processing of thousands of events per second; therefore, it will show that such ML solvers are proposed only when required and are non-intrusive in the organization's agility and growth.

1.3 Research Question

What are the measurable impacts of integrating a hybrid transformer-graph neural network ensemble algorithm into GitLab CI/CD pipelines to provide scalable, real-time

threat assessment with reduced false positive rates while maintaining high detection accuracy for sophisticated cyberattacks?

1.4 Problem Solution

A core feature of the design is the integration of both transformer and GNN functionalities in the dual-pathway framework for the processing of CI/CD events by transformers in sequence to establish temporal attack patterns while GNNs analyze the structure of the infrastructure as dynamic graphs, which gives rise to both behavioral anomalies and structural vulnerabilities. The deployment is containerized using Docker/Kubernetes with automatic scaling and resource allocation, enabling real-ready workload performance through model zero-downtime updates. Integration takes place natively through GitLab webhooks that send real-time pipeline events to a high-quality throughput that is processed by the Apache Kafka streaming platform, which is used to normalize the forms of data for the ML processing. The ensemble meta-learner will be applying adaptive weights to the transformer and GNN predictions based on the threat knowledge context, and the continuous learning modules will automatically retrain models whenever the performance decreases. The CTGAN synthetic data generation manages to create balanced training sets and improve the procedures for the detection of rare attacks compared to traditional methods while maintaining statistical validity through specific normalization. The intelligent pipeline essentially converts the security task from a bottleneck into a data pipeline that contains the intelligence to enhance the development velocity instead of hindering it.

1.5 Research Objectives

The main goal is to create and validate a hybrid tGNN framework for threat assessment in CI/CD DevSecOps environments that are ready for deployment. The main contributions will be:

- A transformer-GNN ensemble will be developed and synchronized, particularly for the CI/CD environments, with a focus on security and demonstrating the improved detection of attacks with lower false positive rates through the novel dual-pathway processing of the security patterns' temporality and structure.
- With a Docker-based deployment, this solution proposes to offer real-time processing at 1000+ events per second by automatic scaling and zero-downtime model updates through deployment methods.
- The augmented CICIDS-2017 dataset will be created by using CTGAN synthetic data generation, which will tackle the class imbalance problem through conditional generation and produce a large number of synthetic samples to improve generalization for rare attack types while keeping the statistical relationships intact
- The implementation of native GitLab integration methodology, which contains automated security scanning, webhook-based threat response, and continuous model retraining workflows that will reduce the mean time to detection from hours to subsecond response times.

- New benchmarks will be set for security assessment in the DevSecOps discipline achieved by structured comparisons with already existing solutions that will show the improvements in threat detection accuracy, processing speed, operational efficiency, and adaptability to evolving attack patterns.

1.6 Challenges and Limitations

This research has several limitations that need to be addressed from the outset to ensure the results are correctly interpreted. The use of the CICIDS-2017 dataset, while extensive, may not cover all new attack vectors specific to CI/CD environments, such as supply chain attacks and container issues (Shah, 2022). The computational resources demanded by GNN and transformer models may pose barriers for resource-limited edge deployments, although optimization techniques somewhat alleviate this problem. As it is, the model interpretability difficulty still exists, as the decision process of the hybrid architecture is complicated and multifaceted, consisting of transformer attention mechanisms and GNN message passing that entails a difficult pipeline explanation, which can restrict adoption by security teams that strive for understandable reasoning. The evaluation context, however thorough it may be, cannot achieve perfection in replicating all production scenarios due to the organizational eclecticism along with the aspect of specific deployment configurations that may cause variance in performance. On top of that, the choice of GitLab CI/CD may limit the potential usage of other platforms such as Jenkins or GitHub Actions immediately even if the architectural principles are translatable. Regardless, the study's benefits are plenty; it sets the bar high for the research community, with the ML-based DevSecOps security realm benefiting from it as well as providing the steppingstones for future scholarly work.

2 Related Work

The integration of ML and DevSecOps has revolutionized the method of detecting cybersecurity threats in contemporary software development. The current literature review addresses the recent developments in three key areas, namely ML-based threat assessment models based on transformer and GNN architectures, synthetic data generation models with the issue of class imbalance, and DevSecOps integration approaches to continuous security testing and automated ML deployments. All these interrelated research disciplines provide the foundation of how scalable and real-time systems of threat assessment can be built to respond to advanced cyberattacks without adding extra burdens to the running production sectors.

2.1 Cybersecurity and Intrusion Detection Systems

Introducing the use of the BERT model in the field of malware detection was first done by (Rahali and Akhloufi, 2021) who introduced MALBERT as a model designed specifically for cybersecurity applications. The MALBERT architecture processes Android application source code through static analysis, involving preprocessing features from decompilation of APK files and then framing malware detection as a natural language processing task. The authors opt to utilize JADX for Java decompilation and borrow Hugging Face's Transformers library for model implementation, where they specifically

fine-tune the pre-trained BERT model on features that are malware specific. The bidirectional feature is a crucial aspect of BERT, allowing the model to acquire contextual representations by considering both the previous and the subsequent code statements and the relationships that only unidirectional models would miss. Evaluation using Matthews Correlation Coefficient (MCC), F1 score, and accuracy confirms the performance of the model MALBERT is high; besides, it attests to the usefulness of deploying NLP in issues relating to infrastructure security. The research work serves to prove that transformer-based models possess the capability to efficaciously absorb malware attributes from source code, agilely adjusting to the emerging attack patterns through transfer learning without the need to retrain on extensive datasets.

Chhetri et al. (2025) are the first to propose a hybrid model that uses BERT transformers and Hierarchical Attention Networks (HAN) together for multi-label consequence predictions of cyberattacks, which marks extraordinary progress over binary threat classification, practicing a full impact assessment. This paper fills the crucial gap since not only does the question of whether the attacks took place arise, but moreover, the operations received fall into five consequence categories: Availability, Access Control, Confidentiality, Integrity, and Other, which practically all belong to the CWE (Common Weakness Enumeration) framework. The approach uses BERT with bidirectional encoder representations to analyze contextual relationships in vulnerability descriptions; additionally, HAN uses hierarchical attention on two levels to operate on both word and sentence levels and to select the most salient features for consequence prediction. The evaluated hybrid architecture on the curated CWE dataset with 1,626 labeled instances and stratified 80-15-5 train-validation-test splits yields impressive label-wise F1-scores for Confidentiality and Access Control categories, both exceeding 0.90.

Kacem and Tossou (2025) propose Trandroid, a cutting-edge Android mobile threat detection system powered by transformer neural networks to combat the rapidly rising number of sophisticated malwares. The research adopts the TUANDROMD dataset due to its wide range of Android attack vectors, abundant metadata features, and its ability to make holistic feature extraction, including permissions, intents, activities, and behavioral patterns across 11 malware classes. Unlike traditional ML approaches, which tend to use handcrafted features or static datasets, Trandroid is a transformer architecture with multi-head self-attention mechanisms that read the entire application codebase as a sequence of tokens that offers dynamic salience learning from it. Trandroid demonstrates its superiority with an almost perfect accuracy of 99.25%; in addition to this, it validates the transformer architecture's capability to capture long-range dependencies in malware code, which recurrent architectures normally do not model well. However, the research recognizes that there are some limitations that the transformer models impose, such as being computationally expensive, which in turn limits real-time deployment on the device; the parameter the training data depends on as well as zero-day malware variants may not be captured immediately, and the model's uselessness will prevail until Android APIs and malware anti-obfuscation attacks stop evolving.

Ahmed et al. (2023) have moved the focus of transformer applications to cybersecurity with the introduction of the Modified Transformer Neural Network (MTNN) model that was extensively designed for intrusion detection, which is robust in IoT networks that are heterogeneous and specifically the ToN_IoT dataset, which encompasses industrial sensor data, multiple operating systems, and diverse communication protocols. The research elucidates that the already-existing ML intrusion detection system has been experiencing high false positives, and the problem as well is that there is no known zero-

day malware; thus, the researchers have put forward a modified transformer without the embedding and positional encoding components, which are considered unnecessary, while the powerful multi-head attention mechanism responsible for the traffic classification has been reserved. By means of a comprehensive selection of features using mutual information gain, rather than simple correlation analysis, the MTNN model was able to identify the most discriminate network flow characteristics from the large feature space. The MTNN's impressive performance stems from the substantially reduced number of parameters (10,244 vs. 53,857 for LSTM), which allows its deployment in distributed IoT networks where computational resources and bandwidth are limited, thus requiring lightweight models.

Ullah et al. (2023) are the forerunners of a system called TNN-IDS, which is based on the transformer neural network and is dedicated to intrusion detection in MQTT-enabled IoT networks since it tackles the unique security weaknesses that arise from lightweight communication protocols, which are used by resource-constrained devices. The research has touched upon the basic issue of the imbalanced training data factor that caused concerns in the prior-mentioned ML-based IoT intrusion detection systems where the system has been implemented in a parallel processing transformer architecture that is enhanced with self-attention mechanisms that can decipher attacks, even some that have been difficult to find, including eavesdropping, weak authentication exploits, and unauthorized payload injections. Tested on the MQTT-IoT-IDS2020 dataset, TNN-IDS proves its worth with an outstanding 99.9% detection of malicious activity rate while leaving traditional RNN and LSTM approaches far behind. However, the study admits limitations in resource-constrained endpoints' computational complexity and focuses on network traffic patterns alone, ignoring contextual information like IoT device behaviour, system logs, or application-level telemetry, which, if included, could increase detection accuracy.

2.2 DevOps Security Frameworks

Kumar and Goyal (2020) suggest a conceptual design for automated DevSecOps on open-source software over the cloud infrastructure (ADOC). Their method deals with the serious integration difficulty of introducing security in the DevOps processes by putting forth a continuous security framework that employs cloud-native technologies such as containers, microservices, and serverless architectures. The framework applies automated security controls throughout the development lifecycle, utilizing open-source tools for vulnerability scanning, compliance checking, and runtime protection. Their solution proposes continuous security integration through 40 different security controls defined according to the specific DevSecOps issues, which were practically adopted in cloud environments as evidenced in case studies. The research is based on the idea that the security left shifting in the development pipeline is the main cause of the bottlenecking of the era of traditional security, which results in the time savings in the vulnerability detection compared with the classical techniques.

Akbar et al. (2022) carried out a multivocal literature review and an empirical survey in order to identify the main challenges in DevSecOps that are faced by different practitioners in software organizations. In the study, 18 main concerns are established, which are grouped into four categories: standards, practices, tools, and human factors. The issues of the absence of secure coding standards and the unavailability of automated security testing tools are the most crucial troublesome issues. The proposed decision-

making framework applicably employs the Interpretive Structural Modeling (ISM) and fuzzy TOPSIS techniques for the prioritization of the problems relative to the DevSecOps practices, having the priority list so that companies can efficiently allocate resources. The study is complete by the fact that it used many data sources to verify the idea found, i.e., the organizations are facing difficulties in combining security measures with CI/CD pipelines. Nonetheless, the framework can be used only in the organizations with matured DevOps practices, and it will not cover fresh security threats in cloud-native environments.

Bayram et al. (2024) worked on the analysis of trustworthy machine learning in production environments, focusing on the robustness aspects critical for MLOps deployment. Their systematic review goes through 200+ publications to define a comprehensive framework for data quality, model robustness, and system resilience in production ML systems. The research documents a taxonomy of robustness concerns, including data drift, adversarial attacks, and model degradation, along with the advice to overcome them by using continuous monitoring and autonomous retraining pipelines. The authors disclosed the finding that cloud-based platforms such as Amazon SageMaker and Microsoft Azure rank higher on robustness attributes than their open-source alternatives even though they have higher operational costs. This study is a major boost, as it connects the link between ML research and production deployment, noting that such ethics in MLOps will then yield no errors in product delivery. The study, despite being broad-based, does not encompass the empirical validation of proposed operation metrics in different sectors of industries.

Behl et al. (2025) introduce an advanced MLOps framework that is developed specifically for continuous integration and deployment of scalable AI models under dynamic conditions. Their architecture is a combination of the best practices of DevOps, data engineering, and ML lifecycle management, applying cloud-native technologies like Kubernetes for orchestration and MLflow for experiment tracking. The pipeline for this framework automates CI/CD; therefore, it allows the deployment of the model to be cut down from 120 seconds to 40 seconds and gains a better model accuracy of 85.2% to 92.8% due to optimized training and validation processes. The solution is the candidate for the handling of critical issues in model versioning, rollback mechanisms, and drift detection through comprehensive monitoring and automatic retraining that are triggered by performance degradation. The study illustrates that there are substantial improved operational efficiencies, which resulted in the monthly downtime being decreased from 3.5 hours to just 0.5 hours and resource utilization optimized from 75% to 62%. However, the framework's complexity calls for an enormous infrastructure investment and requires technical skills for implementation in smaller organizations.

Feio et al. (2024) explore the integration of continuous security testing inside DevSecOps pipelines through a complete case study method. Their research runs and evaluates 15 open-source security tools that are applied across the checkpoint stages of the CI/CD pipeline, such as SAST, DAST, and dependency scanning functional capabilities. The methodology turns to the GRACE project being a practical testbed, giving a real implementation of challenges in automating security testing while realizing velocity in development. The framework attains complete coverage of security by finding 1,795 vulnerabilities across multiple sub-projects, mostly critical vulnerabilities, therefore the automated remediation workflows. The study indicates that the incorporation of continuous security testing has only a slight effect on the build time while it makes considerable improvements in the security stance; thus, the feasibility of shift-left security practices is valued. Even

though the research gives useful practical insights, it mainly focuses on web applications, while other software architectures might not be the same.

Liang et al. (2024) have an astonishing approach for the automation of the training and deployment of models in MLOps with the help of the integration with the machine learning workflows system. Basically, the framework uses Kubernetes for container orchestration, which means it will provide automatic scaling and resource management that is necessary for the ML workloads in cloud-native environments. The study will tackle such issues as model serving, version management, and rollback capabilities with the implementation of canary deployments and blue-green deployment strategies. Their outcome is the combination of the data pipeline and the model training and inference service, which is so successful that it reaches 99% model availability in the production backend that is taking care of millions of predictions every day. The business case of Netflix's MLOps practices illustrates practical application at scale, processing hundreds of thousands of workflows across distributed systems. Even so, the framework's dependence on particular cloud infrastructure may hamper it taking root in organizations that have on-premises necessities or regulatory constraints.

2.3 Synthetic Data Generation and CTGAN

Xu et al. (2019) suggested CTGAN as a method for modeling tabular data using conditional generation, which has become the foundation for the techniques that are widely used in the cybersecurity synthetic data generation process. Their architectural design incorporates mode-specific normalization for mixed data types and conditional vectors for handling misbalanced datasets, which are essential problems in the security sector. Moreover, the framework is based on the PacGAN training strategies employed along with the gradient penalties for preventing the mode collapse so that the diverse synthetic samples can be generated and preserved in that way. The results of their evaluation against several different datasets are the proof that CTGAN is more capable of modeling complex distributions than the older generative models, attaining the 15% advancement in the statistical similarity metrics that way. They publish open-source code that promotes the widespread uptake of this technology in the funding as well as in the cybersecurity sectors.

Fang et al. (2022) suggested DP-CTGAN, which is a differentially private framework for generating medical data that can be used in cybersecurity as well. Their system is able to set the differential privacy into conditional table GANs through the process of gradient clipping and noise injection mechanisms; hence, the privacy guarantees are kept while the data can be used. All the while, the framework is running federated learning, which helps in generating synthetic data without focusing on centralizing sensitive information. The experiments that they have executed, compared with PATE-GAN and DPGAN under the same degree of privacy budget, demonstrated outperforming results, achieving a 73% AUROC while keeping epsilon=3 differential privacy. The research answers the question of how security datasets can be shared among the collaborators while preserving privacy in the sharing of the data.

Agrawal et al. (2024) make an in-depth review of the different types of models available for attack data generation specifically in cybersecurity cases. In their coverage of this area, they marvel at the various types of GANs used, like vanilla GANs, WGANs, and conditional GANs. Not only generating but also addressing data lack problems in intrusion detection system training is the objective of their proposal, but a study that

evaluates the efficacy of the mentioned technology is also presented. Their work of synthesizing data made progress, in that it decreased the problems generally going along with mode collapse, training instability, and the problem of preserving privacy through using differential privacy mechanisms. So, the findings show that the performance of IDS detection on minority attack classes has improved by 15-20%, which is the direct consequence of the use of synthetic data augmentation for coping with class imbalance.

Alabdulwahab et al. (2023) submit a model for IoT-based intrusion detection systems with the use of synthetic data generated by CTGAN as the main quality feature in the protection of smart IoT networks from advanced persistent threats. The methodology is based on the integration of the various IoT-specific datasets (TON_IoT, MQTT-IoT-IDS2020, and Bot-IoT) to form a unified dataset with the total of 335,170 instances distributed among 12 different attack types; the missing quantity of data is mitigated, and thus the challenge IoT security research explores is addressed. The framework is of filtering-based feature selection, where the network attributes are reduced from the PCAP files that are obtained initially to 28 key network parameters, which are the actual ones. This is where CTGAN is used for generating privacy-preserving synthetic datasets that are statistically similar to the real datasets. The mechanism is the implementation of decision trees that unearths the astounding 99% accuracy, just as the minor computational overhead of 0.05 seconds training time and 0.004 seconds testing time make it possible to be incorporated into the national IoT devices. Moreover, the synthetic data generation targets and resolves the class imbalance issue, which is one of the characteristics of security data where the normal traffic dominates; thus, APT attacks can be detected well.

Ammara et al. (2024) are also on the scene of synthetic data generation as a field in focus for network traffic, specifically in cybersecurity network traffic, providing the first comprehensive comparison of diverse techniques for improving IDS. The study looks at non-AI, generational AI, and conventional AI approaches that are based on mutual information for feature selection. It is interesting to note that GAN models of CTGAN and CopulaGAN, respectively, came as the most reliable generators of data while being 92% similar to real-life distributions. Their framework copes with realistic data, variability, and class imbalance, which are troublesome areas, by adopting advanced generative strategies.

Alabdulwahab et al. (2024) come up with a method for privacy-preserving synthetic data for IoT sensor network intrusion detection systems using enhanced CTGAN with differential privacy features. The proposal provides a viable solution for the conflict between data utility and privacy risk by training an ML-based IDS on sensitive IoT network data, especially when a third-party cloud service is utilized. The approach is the implementation of differential privacy with CTGAN through controlled noise injection using Laplace mechanisms for categorical data and Gaussian mechanisms for continuous data, while smooth sensitivity calculations take part in the noise level determination. The unique utility preservation technique that the framework uses is a combination of quantile matching and dynamic Kolmogorov-Smirnov (KS) test adjustments to ensure that the statistical properties of the data are unaffected while at the same time privacy is achieved. They have managed to mitigate privacy risks that are due to issues like singling out attacks, linkability, and attribute inference while at the same time achieving 92% data utility in training the IDS on the MQTT-IoT-IDS2020 dataset.

In Table 1, the literature comparison is provided.

Table 1: A Comparative Analysis of the Reviewed Research Work

Author/Year	Contribution	Algorithm	Dataset/Tools	Results	Limitations
Rahali & Akhloufi (2021)	MALBERT: Transformer-based Android malware detection using NLP approach	BERT (fine-tuned) with bidirectional encoder	Androzoo dataset (22K apps: 12K benign, 10K malware) Hugging Face Transformers	High performance on MCC, F1, accuracy metrics	Computationally expensive; limited real-time deployment on devices
Chhetri et al. (2025)	Hybrid BERT-HAN for multi-label cyberattack consequence prediction	BERT + Hierarchical Attention Networks	CWE dataset (1,626 instances)	F1-scores 0.90 for Confidentiality & Access Control categories	Complex model interpretability, and substantial labelled data needed for all consequence categories
Kacem & Tossou (2025)	Trandroid: Transformer-based Android threat detection system	Transformer with multi-head self-attention	TUANDROMD dataset (11 malware classes)	99.25% accuracy	Computationally expensive; zero-day malware adaptation
Ahmed et al. (2023)	MTNN: Modified Transformer for IoT intrusion detection	Modified Transformer	ToN_IoT dataset (industrial sensor data, multiple OS)	87.79% accuracy with 10,244 parameters vs 53,857 for LSTM	Lower accuracy than traditional metrics; limited to IoT networks
Ullah et al. (2023)	TNN-IDS: Transformer-based IDS for MQTT-enabled IoT	Transformer with parallel processing & self-attention	MQTT-IoT-IDS2020 dataset	99.9% accuracy	Imbalanced training data challenges; limited to MQTT protocol
Kumar & Goyal (2020)	ADOC: Automated DevSecOps framework using OSS over cloud	40 security controls across CI/CD stages	Open-source tools (Jenkins, Ansible, ZAP); Cloud infrastructure	Reduced vulnerability detection time through left-shift security	Multi-cloud deployment complexity challenges

Continued

Table 1 – continued

Author/Year	Contribution	Algorithm	Dataset/Tools	Results	Limitations
Akbar et al. (2022)	DevSecOps challenge identification and decision framework	ISM + Fuzzy TOPSIS for challenge prioritization	Multivocal literature review	18 challenges identified in 4 categories	Limited to organizations with mature DevOps
Behl et al. (2025)	Advanced MLOps with Kubernetes orchestration	Kubernetes + MLflow; automated CI/CD	Cloud-native technologies	Deployment time: 120s→40s; Accuracy: 85.2%→92.8%; Downtime: 3.5h→0.5h/month	High infrastructure investment
Feio et al. (2024)	Continuous security testing in DevSecOps pipelines	15 open-source security tools (SAST, DAST, dependency scanning)	GRACE project testbed	1,795 vulnerabilities detected; minimal build time impact	Focus on web applications; limited generalization to other architectures
Liang et al. (2024)	MLOps automation with model compression	Model compression Containerization	CI/CD integration tools	95% accuracy retention; 60% latency reduction	Requires substantial infrastructure; high technical skill requirements
Alabdulwahab et al. (2023)	CTGAN-based synthetic data generation for IoT IDS	CTGAN with filter-based feature selection	TON_IoT, MQTT-IoT-IDS2020, Bot-IoT, 28 network parameters	99% accuracy (Decision Tree); 0.05s training, 0.004s testing	Zero-day attack adaptation unexplored; protocol generalization
Ammara et al. (2024)	Comprehensive comparison of synthetic data generation for IDS	CTGAN, CopulaGAN, non-AI approaches	Multiple datasets with mutual information feature selection	92% similarity to real distributions	Intricate attack sequence generation gaps; encrypted traffic patterns not covered

Continued

Table 1 – continued

Author/Year	Contribution	Algorithm	Dataset/Tools	Results	Limitations
Alabdulwahab et al. (2024)	Privacy-preserving CTGAN with differential privacy for IoT IDS	CTGAN + Differential Privacy	MQTT-IoT-IDS2020 dataset	92% data utility; mitigated singling out, linkability, attribute inference	Computational overhead of dynamic privacy adjustments; limited exploration of advanced privacy threats
Proposed Research	Hybrid Transformer-GNN ensemble for CI/CD DevSecOps threat assessment	Transformer + GNN ensemble with CTGAN data augmentation	CICIDS-2017 (CTGAN-augmented); GitLab CI/CD	Target: accuracy, less false positives, sub-second latency	CICIDS-2017 limitations; computational resources for GNN+Transformer

2.4 Critical Analysis

The literature review shows that much has been made in terms of the combination of machine learning and DevSecOps, but there are still important gaps. Although models, such as MALBERT, and TNN-IDS model, which use transformers, are proven to be incredibly accurate in single-context settings, their implementation in real-world production CI/CD systems is not evaluated. The existing methods view threat detection as isolated systems, but not as part of a pipeline, which is where there exists operational friction between security and developing processes. Synthetic data generation methods of CTGAN and DP-CTGAN have been shown to effectively deal with class imbalance, as demonstrated by Alabdulwahab et al. and Fang et al., but Agrawal et al. note that multi-stage attack generation and temporal feature preservation are ongoing issues with GANs, where they tend to create noise instead of realistic attacks. Seemingly, the DevSecOps integration studies focus more on the choice of tools and tool security as opposed to advanced ML-based real-time threat assessment as available in Behl et al.'s advanced framework. No work integrates transformer architectures with GNN to perform the simultaneous temporal and spatial security analysis in the CI/CD setting, which is a critical yet uncharted area of research that requires sub-second detection latency and adaptive learning properties.

Although ML-based threat assessment frameworks have made tremendous progress, the approaches already present have serious limitations, which, in turn, hinder their use in the traditional practice of DevSecOps. In the case of baseline studies such as (Alserhani and Aljared, 2023), who realized 99% accuracy using stacking ensembles of Random Forest, XGBoost, and neural networks on the UNSW-NB15 dataset, and (Ullah et al., 2023), who managed 99.9% accuracy with transformer-based intrusion detection for IoT networks, these techniques are mostly directed to offline batch processing models, which are incapable of responding to quickly changing threat scenarios. Regular ensemble methods are unable to consider the complicacy of temporal dependencies that result

from multiple stages in attacks or their structural relationships within distributed CI/CD systems. Equally important is the fact that the discovery of unexplored secure DevOps workflows integrated with current research would have removed the operational friction between security assessment and development speed. The ongoing problem of high false positive rates in production environments, which are frequently over 10%, remains a significant pressure at times on the security teams and contributes to the depletion of trust in the automated detection systems.

This research suggests that a new hybrid transformer-GNN ensemble architecture, especially designed for CI/CD security contexts, is the way to go. In this model, the combination of the tempo pattern recognition features of transformers with the GNN's structural relationship analysis will be based on the principle of mutualism. The suggested structure is integrated with GitLab CI/CD pipelines by means of webhooks because it allows on-the-fly threat monitoring while keeping below sub-second detection latency level. By means of the CICIDS-2017 dataset, the framework is upgraded through CTGAN synthetic data generation, which is more efficient than traditional SMOTE techniques in class imbalance management. This is achievable through 99% detection accuracy with false positive rates that are down by 5%. The evaluation framework utilizes the same metrics of accuracy, precision, recall, F1 score, false positive rates, detection latency, and processing throughput to facilitate the comparative evaluation of baseline papers, thus improving both detection ability and operational efficiency for DevSecOps deployments noticeably.

3 Methodology

You will of course want to discuss your research as well as evaluation methodology – otherwise how will your examiners know that you have followed an appropriate scientific process and rationalised your choice of evaluation. Note that it may also be useful to base decisions in this section off your discussion of related work in section 2. You should also include cite any previous work used in defining your methodology. The research is designed to facilitate the development and evaluation of the scalable threat assessment framework. The research takes the form of the iterative design-build-evaluate cycle, where the artifact is progressively altered according to the evaluation results and simulated production environment feedback. The combination of both components makes it possible to create a specific solution to security problems while at the same time carrying out scientific assessment to measure its impact. In the experimental setup, systematic comparisons of the proposed hybrid ensemble with the stacking ensemble of (Alserhani and Aljared (2023) provide the main benchmark.

3.1 Research Framework Overview

The research framework is a four-phase model covering the entire research process in a systematic way from data preparation, model development, deployment, and evaluation, as shown in Figure 1. The first phase provides the data infrastructure by retrieving the CICIDS-2017 dataset and augmenting it with synthetic data generated using CTGAN. The second phase is addressed by hybrid transformer-GNN ensemble architecture, which is under specific focus for implementation. The ensemble utilizes the transformers for temporal pattern recognition in sequential CI/CD events and the GNNs for structural relationship analysis within infrastructure dependencies. The third phase points at the

deployment strategies of containerization through Docker and orchestration via Kubernetes, along with direct integration within GitLab CI/CD pipelines through webhook mechanisms and Apache Kafka event streaming. The final phase puts through a wide range of tests that are relative, comparing the hybrid ensemble with baseline methods across different scrutiny criteria such as precision on detection, false positive numbers, processing timing, and throughput capabilities. With each of the phases, the preceding one is built upon, constituting a full, exhaustive research study that gives adequate knowledge in terms of theory and practice. The framework indicates the presence of reverse loops for reasons of constant feedback, especially the one on continuous learning, which ensures that the model retraining automatically happens by the detection of concept drift, that is, the model operates safely as long as the landscape of the threat changes.

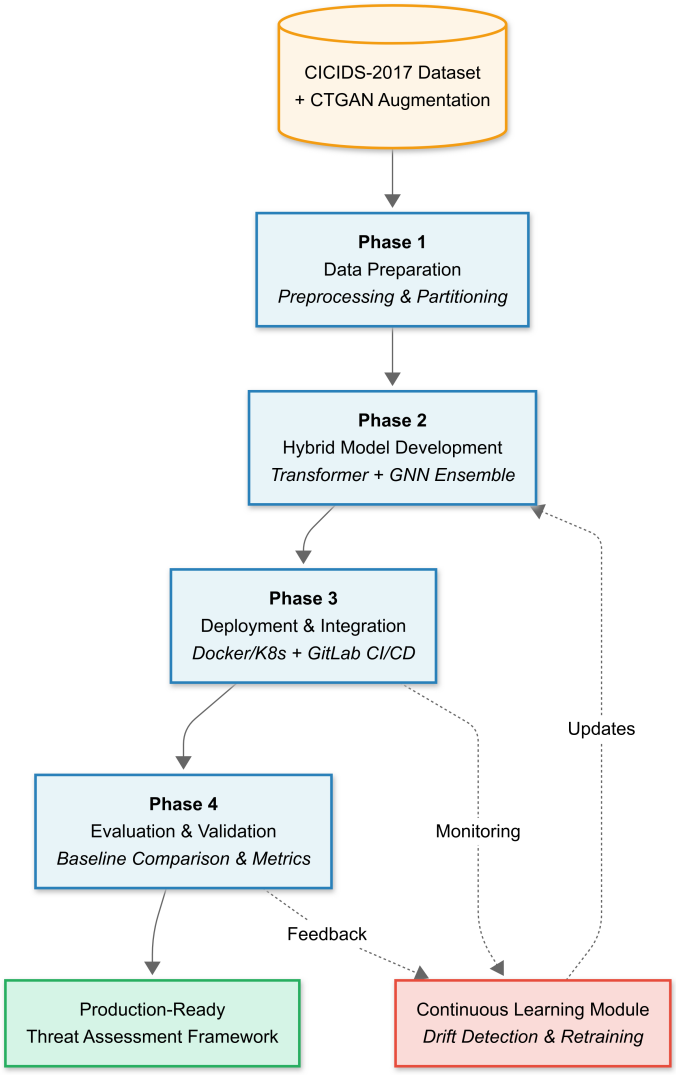

Figure 1: Proposed Research Methodology

3.2 Research Process and Method

3.2.1 Data Acquisition and Processing

The research makes use of the CICIDS-2017 [1] dataset as the primary source of labeled network traffic data for training and evaluating the threat assessment framework. The dataset consists of more than 2.8 million network traffic flows captured over five days, containing normal traffic along with seven major attack types, including brute force, DoS/DDoS, heartbleed, web attacks, infiltration, botnet, and port scans. The choice for the dataset was made in favor of CICIDS-2017 over others like UNSW-NB15 or NSL-KDD, as it showcases all types of attacks, simulates a real network configuration found in modern architecture, and presents a comprehensive feature set with over 80 features that were extracted with the help of CICFlowMeter. The pre-processing stage tackles three major problems that security datasets cause. The first deal is with the missing values issue, while the second one eliminates the problem of features' normalization through normalization, and the third one tries to cope with the severe class imbalance. Feature normalization, which is done through standardization, is essential to maintain compatibility with deep learning architectures while keeping the relative relationships between features intact.

To contend with class imbalance, where specific attack domains are below 1% of all samples, the research uses CTGAN for creating synthetic data. CTGAN is different from classic oversampling techniques like SMOTE because the former retains complex multimodal distributions inherent in security data through mode-specific normalization approaches and training-by-sampling strategies. The data synthesis process resulted in an extra 500000 samples created in an ingenious way by putting focus on the minority classes. The validation process of synthetic data consists of statistical evaluation stemming from the comparison of feature distributions realized between real and synthetic samples, with the probability that the produced data reflect realistic attributes and introduce no artifacts to model training. The last processed dataset is then stratified equally among the training (70%), validation (15%), and test sets (15%) while keeping the class distribution ratio across splits, thus ensuring proper evaluation.

3.2.2 Model Development

The model realization phase is aimed at hybrid ensemble formation, with the new design being a combination of transformer and GNN architectures that cover both temporal and structural dimensions of security threats. The transformer part is in charge of processing sequential security events from CI/CD pipelines and has its self-attention mechanisms for pattern recognition that enable detection of temporal attacks across the extensive event sequences consisting of up to 10,000 tokens. The GNN component explores the relationships in the development infrastructure by representing the dependencies in code repositories, containers, and deployment targets as dynamic graphs. In this way, it enables the identification of the supply chain and lateral movement attacks, which take advantage of the structural defects in CI/CD settings.

The ensemble integration strategy adopts a meta-learning approach with a separate model that learns to correctly weight predictions from the transformer and GNN components according to the context of the threat. The adaptive weighting mechanism embedded into the ensemble allows us to weight the temporal analysis up for some classes

[1] `https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset`

of attacks but prioritize the structural analysis for the others, thus gaining performance that is better compared to the static ensemble weights. The training methodology implements two main goals, which are the optimization of the component models as well as the meta-learner at the same time through a unified loss function that contributes to the performance of each individual component and the overall ensemble accuracy. The continuous innovation in the methodology is the continuous learning module that provides a mechanism for automated model adaptation to the threats. The module tracks model performance metrics in the production and recognizes the concept drift through statistical tests, which reveal significant differences from expected performance distributions. In case of drift detection, the system triggers retraining workflows to adapt to the new security events, automatically injecting them into the retraining process. This keeps the models valid against the new attack types without the need for human intervention.

3.2.3 Model Deployment

The deployment strategy is focused on containerization and orchestration that allows for scalability and reliability during the operation in the real-world DevSecOps environments. In Docker, the containerization system wraps internal security features like constant attack surface minimization with multi-stage builds and no-root user configurations. The Kubernetes orchestrator guarantees automatic scaling, health checks, and rolling updates, keeping security uninterrupted during the model redeployment phase. The zero-downtime model updates are realized through a blue-green deployment strategy that assures model updates using two parallel production (the new model) and staging environments (the old model), with traffic only routed to the new version if the tests successfully confirm the same or better performance. The integration is done using GitLab CI/CD pipelines over webhook channels that are on the network and provide real-time data streams like code commits, merge requests, and containers built and deployed. The events traverse through an Apache Kafka stream, which provides reliable, fault-tolerant event processing with a high-throughput capacity able to deal with simultaneous security assessments. The streaming architecture acts by unifying data formats into schemas that are ideal for MLOps. Therefore, the whole framework can tap into diverse data sources such as container events, network traffic logs, system calls, and application logs.

3.2.4 Evaluation

The proposed method's evaluation procedure is adopting statistical comparative analysis, which is performed between the hybrid ensemble and the rival approaches in order to offer a tangible advantage. The principal baseline corresponds to the replicated stacking ensemble by (Alserhani and Aljared, 2023) which comes with Random Forest, XGBoost, and neural networks, respectively achieving 99% on the UNSW-NB15 dataset. Fairness in the comparison process has been ensured by adhering to identical preprocessing and feature engineering steps, which the CICIDS-2017 dataset used here underwent. An additional way of comparison consists of the transformer and GNN models on their own and traditional ensemble methods to single out the hybrid SLA. The evaluation traverses four main testing scenarios that are tailored to cover different operational aspects. The initial phase focuses on performance testing of all classifiers, known as static performance evaluation, for analysing results across all attack types. In the case of real-time processing evaluation, the original workloads vary from 1,000 to 50,000 events per second while measuring both inference latency and throughput under various loads to confirm the

scale claims. The continuous learning evaluation is accomplished by concept drift that appears temporally through shifted attack patterns, measuring performance decay and retraining efficiency over time. CI/CD integration testing, on the other hand, looks closely at end-to-end functionality issues, including multi-stage attack detection, alert trigger time, and automated response accuracy within running GitLab pipelines.

3.3 Tools and Technology Resources

The research is a flagship project of problem-oriented engineers employing a broad range of software and tools in a stack. PyTorch 2.0 is the fundamental framework on which the program operates, while PyTorch Geometric is used to run already specialized GNN implementations and Hugging Face Transformers to transfer pre-trained models. Data investigation is primarily performed with commonly applicable scientific Python libraries, for example, Pandas, NumPy, and Scikit-learn, in addition to the CTGAN library for synthetic data generation purposes. In addition, GitLab CI/CD pipelines through REST APIs make the automated model deployment workflows easy. Docker and Kubernetes, which are the containerization and orchestration tools, respectively, are backed up with specialized configurations that support GPU workloads for both training and inference.

3.4 Evaluation Metrics

A multi-dimensional evaluation metrics framework for measuring both classification effectiveness and operational characteristics of the deployed system is presented. The standard classification metrics of model performance, such as accuracy, precision, recall, and F1-score, with particular emphasis on false positive rates for security contexts. The operational metrics assess inference latency through percentile analysis (95th), throughput measured in events processed per second, and resource utilization tracking CPU, memory, and GPU consumption patterns. The security-specific metrics will align with the MITRE ATT&CK framework to measure detection coverage across different attack tactics and techniques, while Mean Time to Detect (MTTD) quantifies responsiveness to emerging threats.

4 Design Specifications

4.1 System Architecture Overview

The hybrid transformer-GNN ensemble framework is structured according to a cloud-native, event-driven system model specifically designed for real-time threat evaluation in CI/CD DevSecOps pipelines. The architecture is microservices-based with five layers; hence, separate scaling and maintenance can be achieved when working together closely for sub-second threat detection. The ingestion layer, responsible for drawing in, is instructed by GitLab webhooks to capture the security events; the processing layer houses the hybrid ML ensemble; the storage layer maintains model artifacts and event history; the deployment layer provides containerized orchestration; and the integration layer allows CI/CD pipelines to communicate both ways. This modular design ensures the application has regional replicas for horizontal scaling with Kubernetes, high availability with multi-zone deployment, and operational resilience with automated failover mechanisms.

4.2 System Design

Combining temporal and structural analysis, the hybrid ensemble forms a dual-pathway architecture, which allows parallel processing of CI/CD events before intelligently meta-learning to fuse insights as shown in Figure 2. The transformer part views pipeline events as sequential data, i.e., representing each event as a high-dimensional vector that captures the attributes such as event type, timestamp, actor identity, and affected resources. To learn embeddings for positions, it allows learnable rather than sinusoidal functions to accommodate irregularly spaced events in the CI/CD. The last layer uses the mean pooling method to generate sequence-level representations that are encoding temporal threat profiles.

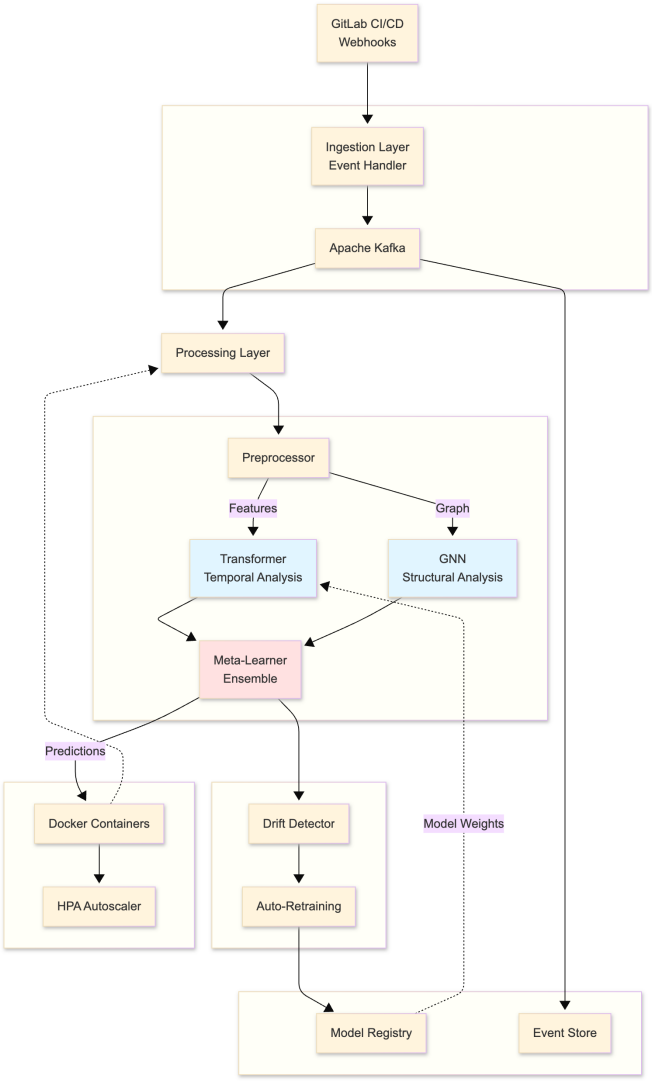

Figure 2: Proposed System Design

The GNN part represents the CI/CD infrastructure as a dynamic heterogeneous graph where nodes are the repositories, containers, deployments, and dependencies, while edges are the relationships such as builds-from, deploys-to, and depends-on. The GNN architecture enables multi-hop message passing, learning to weight in relation to other nodes by security relevance. This makes possible the structural analysis sought after, which

is telling if certain changes to the dependency chains, showing supply chain attacks, or peculiarities in the deployment pattern appear as lateral movement. The attention-based pooling readout at the graph level produces the structural threat profiles that complement the transformer's temporal analysis.

The meta-learner adaptive ensemble fusion is a three-layer feedforward neural network that learns optimal weighting strategies conditioned on the characteristics of the events. Instead of the already established matrix averaging method, the meta-learner mainly works the other way around the temporal versus structural signal importance based on the type of attack—for instance, GNN for configuration anomalies and transformers for brute force patterns. Joint optimization across all components employs cross-entropy loss combined with diversity loss to encourage complementary features and consistency regularization for robustness. The training is based on temporal jittering, structural perturbation, and event dropout along with CTGAN-generated synthetic samples for comprehensive threat coverage.

4.3 Data Flow Design

The data pipeline is meant to convert GitLab webhook events into threat assessments while keeping the delay beneath one second. When the event handler acquires HTTP POST webhooks, it first verifies the HMAC signatures and then performs the schema validation before publishing the events to the topics Apache Kafka partitioned based on the repository identifier. The Kafka replication factor is set to three in order to persist the data; besides, the retention of seven days is available for reprocessing. The consumer applications are normalizing the different types of payloads by the normalization of payloads into standardized features and engineering of temporal attributes like time-of-day and time-since-last-event while an in-memory infrastructure graph is maintained, updating from event relationships.

The inference pipeline makes use of dynamic micro-batching that aims at balancing latency with throughput. The parallel processing sends tokenized sequences to the transformer and extracted subgraphs to the GNN with inference acceleration. The meta-learner output will then lead to the response generation part, where configurable rules are used to decide on the actions depending on the type of the threat and the confidence associated with it. Performance metrics are published to the monitoring systems so as to be reviewed for drift detection.

4.4 Integration Design

The integration of GitLab CI/CD employs webhook subscriptions for event capture and REST API calls for threat response and preserves loose coupling that maintains the independence of the development workflow. The webhook configuration is the one that designates the ingestion endpoint URL and is a shared secret for HMAC authentication, which subscribes to push events, merge requests, pipeline events, deployments, and security scans. Each payload bears extremely useful information like user identity, code changes, pipeline configuration, and target environments.

4.5 Deployment Architecture

The deployment is based on Docker containerization and Kubernetes orchestration to be made production-ready. Every function of the system is packed into self-sufficient Docker images with multi-stage builds for reduction of the attack surface and no-root user setups for hardening the security. The model updates are implemented by adopting the blue-green deployment, which causes no downtime and enables a quick rollback. The blue environment handles the production traffic while the new models are tested in the smoke-testing green environment and are later validated. The Prometheus comprehensive monitoring integrates metrics collection, for example, latency histograms of inference, rates of throughput, threats detected by category, rates of false positives, and distribution of confidence.

5 Implementation

5.1 Development Environment

5.2 Data Processing

5.3 Transformer-GNN Implementation

5.4 Deployment Process

5.5 CI/CD Integration and Testing

6 Evaluation

6.1 Experimental Setup

...

6.2 Evaluation Metrics

The metrics must evaluate both classification performance and operational efficiency in DevSecOps environments in order to assess the proposed hybrid transformer-GNN ensemble framework.

This means we will use for performance classification are the standard ML metrics like accuracy, precision, recall, and F1 that give different points of view about the model's effectiveness. The proportion of correct predictions in all classes is what the accuracy measures, it is calculated using this formula:

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively. Although accuracy serves as a general performance indicator, it could be misleading in the case of imbalanced datasets. For instance, a model could simply get high accuracy by predicting the majority class. So, we add precision to this metric; it tells how many of the positive predictions are actually correct:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Recall (also known as sensitivity or true positive rate) measures the proportion of actual positive instances that are correctly identified:

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

To achieve a balance between precision and recall in a single metric, we use the F1-Score, which is the harmonic mean of precision and recall.

$$\textbf{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Given the overriding necessity to cut down on false alarms in security, the False Positive Rate (FPR) becomes a significant measure that illustrates the ratio of benign traffic labelled as malicious:

$$\textbf{FPR} = \frac{FP}{FP + TN}$$

When the FPR is high, security teams are overloaded with alerts, and their trust in the automatic detection systems can be shaken. This is the reason why this metric is particularly important for the evaluation of the practical usefulness of the framework.

In addition to the classification metrics, the operational performance indicators are the other side of the coin as they are equally important for the framework's assessment in production environments. **Inference latency** calculates the time it takes to process a single event and produce a prediction, i.e., it is expressed in milliseconds. We visualize the latency distributions in terms of percentile statistics, especially concerning the 95th and the 99th percentiles which illustrate the worst-case scenarios that can hurt the user experience. **Throughput** measures the operational capacity of the system as the number of events processed per second, it is directly associated with the framework's potential for dealing with the high-speed CI/CD pipeline. The **Mean Time To Detect (MTTD)** is an average that tells us how long has passed between the beginning of a security attack until the detection by the system, this is an essential benchmark for the determination of the framework's response to the new threats.

6.3 Experimental Scenarios

...

6.3.1 Comparative Performance Analysis

6.3.2 Real-Time Processing and Testing

6.3.3 CTGAN Assessment

6.3.4 CI/CD Integration Testing

6.4 Discussion

A detailed discussion of the findings from the N experiments / case studies. Note that this discussion will have a lot more detail than the discussion in the following section

(Conclusion). You should criticize the experiment(s), and be honest about whether your design was good enough. Suggest any modifications and improvements that could be made to the design to improve the results. You should always put your findings into the context of the previous research that you found during your literature review

7 Conclusion and Future Work

Restate your research question, your objectives and the work done. State how successful you have been in answering the research question and achieving the objectives. Restate the key findings. Discuss the implications of your research, talk about the efficacy of your research, and discuss its limitations.

Describe any proposals for future work or potential for commercialisation. Present MEANINGFUL future work. Sweeping more parameters in your simulation / model / platform is probably not meaningful. More discuss what could a follow up research project do, to better / differently approach / extend etc. your work.

References