American Journal of Electrical and Computer Engineering
2026, Vol. 10, No. 1, pp. 1–18
https://doi.org/10.11648/j.ajece.20261001.11

SciencePG
Science Publishing Group

Research Article

Machine Learning-Driven Intrusion Detection with Forensic Readiness in Cloud-Native IoT Environments

Frank Idugboe[1, *], Hilary Onyeka Okonkwo[2], Blessing Oboh Ejiyere[2], Esther Ijeoma Ugo[3]

[1]Department of Electrical Engineering, Yaba College of Technology, Lagos, Nigeria

[2]Department of Computer Science, Ambrose Alli University Ekpoma, Edo, Nigeria

[3]Department of Maths/Statistics/Computer Science, Michael Okpara University of Agriculture, Umudike, Nigeria

Abstract

Cloud-native Internet of Things (IoT) environments combine connected devices, edge gateways, containerized services, Kubernetes orchestration, microservices, and cloud infrastructure. Although this architecture improves scalability and automation, it also expands the attack surface and complicates forensic investigation. Conventional intrusion detection systems often focus on detection accuracy but provide limited support for evidence preservation, chain-of-custody management, and incident timeline reconstruction. This study proposes and evaluates a machine learning-driven intrusion detection framework with forensic readiness for cloud-native IoT environments. The CICIoT2023 dataset was used to evaluate Logistic Regression, Decision Tree, Random Forest, XGBoost, and Autoencoder models under binary and multi-class classification settings using an 80: 20 train-test split and 5-fold cross-validation. Experimental results show that XGBoost achieved the best performance. In binary classification, it obtained 99.34% accuracy, 99.35% precision, 99.34% recall, 99.34% F1-score, and 99.89% ROC-AUC. In multi-class classification, it achieved 97.69% accuracy, 96.12% macro precision, 95.07% macro recall, 95.54% macro F1-score, and 97.65% weighted F1-score. The forensic readiness evaluation showed 96.15% Evidence Completeness Ratio, 100.00% Chain-of-Custody Completeness, 100.00% Evidence Integrity Score, 0.84-second average preservation latency, 98.72% Alert-to-Evidence Mapping Rate, 94.60% Timeline Reconstruction Success Rate, and 96.89% Investigation Readiness Index. The findings demonstrate that the proposed framework supports accurate intrusion detection and investigation-ready evidence preservation for cloud-native IoT security.

Keywords

Machine Learning, Intrusion Detection, Forensic Readiness, Cloud-native IoT, CICIoT2023, XGBoost, Digital Forensics, Kubernetes Security

1. Introduction

The Internet of Things (IoT) has become central to modern digital systems, supporting applications in healthcare, smart homes, transportation, industrial automation, agriculture, and critical infrastructure. However, the growing connectivity of

*Correspondence: Frank Idugboe (frank.idugboe@unifesspa.edu.br)

Received: 13 May 2026; **Accepted:** 22 May 2026; **Published:** 29 June 2026

IoT devices has also increased exposure to cyber threats such as distributed denial of service (DDoS), denial of service (DoS), reconnaissance, spoofing, brute force, web-based attacks, and botnet activity. These risks become more complex in cloud-native environments, where IoT devices interact with edge gateways, containers, microservices, Kubernetes clusters, service meshes, cloud APIs, and distributed storage. Cloud-native computing improves scalability, automation, resilience, and observability through technologies such as containers, microservices, service meshes, immutable infrastructure, and declarative APIs [6]. However, these technologies also introduce new security and forensic challenges. Attacks in cloud-native IoT environments may move across multiple layers, including devices, edge nodes, containerized workloads, orchestration platforms, and cloud services. Therefore, intrusion detection must go beyond traditional network monitoring and include cloud-native telemetry such as container logs, Kubernetes audit logs, service-mesh traces, and cloud access records. Machine learning has become an important approach for IoT intrusion detection because it can learn complex traffic patterns and identify abnormal behavior from network-flow features. Datasets such as CICIoT2023 support this research by providing realistic IoT attack traffic. CICIoT2023 contains 33 attacks generated from an IoT topology of 105 devices and grouped into DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai categories [3, 15]. Despite these advances, many machine learning-based IDS studies focus mainly on detection metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Limited attention is given to forensic readiness, including evidence preservation, cryptographic hashing, chain-of-custody records, alert-to-evidence mapping, and incident timeline reconstruction. This is a serious limitation because intrusion alerts must support not only real-time detection but also post-incident investigation. NIST forensic guidance emphasizes the importance of reliable collection, examination, analysis, and reporting of digital evidence during incident response [10].

This study addresses this gap by proposing and evaluating a machine learning-driven intrusion detection framework with forensic readiness for cloud-native IoT environments. The framework integrates IoT network-flow analysis, edge-level detection, cloud-side classification, and forensic evidence management. Five models—Logistic Regression, Decision Tree, Random Forest, XGBoost, and Autoencoder—are evaluated using CICIoT2023 under binary and multi-class classification settings. In addition to conventional IDS metrics, the study evaluates forensic-readiness indicators such as Evidence Completeness Ratio, Chain-of-Custody Completeness, Evidence Integrity Score, Preservation Latency, Alert-to-Evidence Mapping Rate, Timeline Reconstruction Success Rate, and Investigation Readiness Index.

The objectives of this study are to:

1) develop a forensic-ready machine learning-driven IDS framework for cloud-native IoT environments;
2) evaluate selected machine learning models for binary and multi-class IoT intrusion detection;
3) compare model performance using detection and runtime metrics;
4) assess forensic readiness using evidence completeness, integrity, chain-of-custody, preservation latency, and timeline reconstruction metrics; and
5) demonstrate how intrusion alerts can be transformed into investigation-ready evidence packages.

By addressing both detection accuracy and forensic readiness, this study contributes to the development of more practical, accountable, and investigation-ready intrusion detection systems for cloud-native IoT deployments.

2. Literature Review

2.1. Cloud-Native IoT Security Context

The Internet of Things (IoT) has become a major component of modern digital infrastructure, supporting smart homes, healthcare monitoring, industrial automation, smart cities, transportation systems, and critical infrastructure. However, the growth of IoT has also expanded the cyberattack surface because many IoT devices have limited processing power, weak authentication mechanisms, constrained storage, and inconsistent patching practices. These limitations make IoT systems vulnerable to attacks such as distributed denial of service, denial of service, spoofing, brute force, botnet propagation, reconnaissance, and web-based exploitation [11, 13, 15]. The security challenge becomes more complex when IoT systems are deployed in cloud-native environments. Cloud-native systems rely on containers, microservices, service meshes, immutable infrastructure, declarative APIs, Kubernetes orchestration, and elastic cloud services. The Cloud Native Computing Foundation explains that cloud-native technologies support scalable, resilient, manageable, and observable systems through containers, service meshes, microservices, immutable infrastructure, and declarative APIs [6]. While these technologies improve scalability and automation, they also introduce additional security monitoring challenges because attacks may move across IoT devices, edge gateways, containerized services, Kubernetes APIs, cloud workloads, and distributed storage systems.

In cloud-native IoT environments, security monitoring must therefore move beyond conventional network inspection. Traditional network intrusion detection may capture packet-level or flow-level anomalies, but it may not capture Kubernetes audit events, container lifecycle events, service account misuse, pod-to-pod communication, API abuse, or cloud access anomalies. Recent Kubernetes security research shows that cloud-native environments require security approaches that can detect workload-level threats and integrate Kubernetes-specific telemetry into threat analysis [17].

2.2. Intrusion Detection Systems in IoT

Intrusion detection systems are widely used as a second line

of defense to identify suspicious or malicious activity that bypasses preventive controls. In IoT environments, IDSs are especially important because IoT devices are often exposed to public networks, rely on lightweight protocols, and may lack strong endpoint protection. IDS methods are usually classified into signature-based, anomaly-based, specification-based, and machine learning-based approaches [16, 20]. Signature-based IDSs are effective for detecting known attacks but are limited when facing zero-day threats, polymorphic malware, and evolving IoT botnets. Anomaly-based IDSs can detect unknown behavior by modeling normal traffic patterns, but they often suffer from high false-positive rates. Machine learning-based IDSs address some of these limitations by learning complex relationships between traffic features and attack labels.

Recent IDS surveys emphasize that machine learning and deep learning can improve detection accuracy, reduce false alarms, and support dynamic threat identification in IoT networks [16, 21]. IoT IDS research increasingly focuses on flow-based detection because flow records are more scalable than full packet inspection and are easier to process in real-time systems. Flow-level features such as packet count, byte count, flow duration, inter-arrival time, protocol type, source and destination ports, and traffic direction can help machine learning models distinguish benign behavior from malicious traffic. However, flow-based detection also has limitations because some attack categories may share similar statistical characteristics. This is why multi-class IoT intrusion detection remains more difficult than binary detection.

2.3. Machine Learning Techniques for IoT Intrusion Detection

Machine learning has become central to IoT IDS research because it can identify nonlinear relationships in high-dimensional network traffic. Classical models such as Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbors, and Naïve Bayes are frequently used as baselines in IDS studies. Ensemble models such as Random Forest and XGBoost often perform strongly on tabular network-flow datasets because they can capture nonlinear interactions among features while reducing overfitting [2, 4]. Random Forest combines multiple decision trees and uses voting to improve classification robustness. Breiman described random forests as ensembles of tree predictors in which each tree depends on independently sampled random vectors, and the generalization error depends on tree strength and inter-tree correlation [2]. This makes Random Forest suitable for intrusion detection because it handles nonlinear relationships, feature interactions, and noisy data better than a single decision tree. XGBoost is another powerful ensemble method. Chen and Guestrin introduced XGBoost as a scalable tree-boosting system designed for efficient and accurate supervised learning [4]. XGBoost uses gradient boosting, regularization, sparsity-aware learning, and optimized tree construction, making it highly effective for structured datasets such as network-flow intrusion data. Its strong performance in many classification tasks explains why it is frequently used in IDS experiments. Deep learning and anomaly detection approaches have also been applied to IoT IDS. Autoencoders are useful when labeled attack data are limited because they can learn normal behavior and detect deviations from that behavior. Meidan et al. proposed N-BaIoT, a network-based approach using deep autoencoders to detect IoT botnet attacks from compromised devices infected by Mirai and BASHLITE [13]. Their work showed that autoencoder-based anomaly detection can identify abnormal behavior from IoT devices by learning device-specific normal traffic patterns. Isolation Forest is another anomaly detection approach relevant to IDS research. Liu et al. proposed Isolation Forest based on the idea that anomalies are few and different, making them easier to isolate using random partitioning [12]. This method is useful for lightweight anomaly detection because it does not require distance or density estimation, which can be computationally expensive in high-dimensional datasets. Despite the usefulness of deep learning, many studies note that deep models may require more computational resources, larger datasets, and longer training times. This creates deployment challenges for edge-based IoT systems. Therefore, practical cloud-native IoT IDS frameworks often need a layered model strategy: lightweight models at the edge and stronger ensemble or deep learning models at the cloud or edge-gateway layer.

2.4. IoT Intrusion Detection Datasets

Datasets are critical in IDS research because model performance depends heavily on the realism, diversity, and labeling quality of the training data. Older network intrusion datasets are often criticized for being outdated or not representative of modern IoT traffic. For IoT IDS research, commonly used datasets include CICIoT2023, Bot-IoT, TON_IoT, Edge-IIoTSet, and N-BaIoT. CICIoT2023 is one of the most relevant datasets for modern IoT intrusion detection. Neto et al. introduced CICIoT2023 as a real-time dataset and benchmark for large-scale attacks in IoT environments [15]. The dataset was developed using a topology of 105 IoT devices and includes 33 attacks grouped into seven major classes: DDoS, DoS, Reconnaissance, Web-based, Brute Force, Spoofing, and Mirai. The Canadian Institute for Cybersecurity also describes CICIoT2023 as a dataset designed to support security analytics in real IoT operations [3]. Bot-IoT is another important dataset. Koroniotis et al. developed the Bot-IoT dataset to support realistic botnet detection and network forensic analytics in IoT networks [11]. The UNSW dataset page explains that Bot-IoT was developed for botnet activities and network forensic analytics, making it particularly relevant to studies that connect intrusion detection with forensic analysis. TON_IoT was introduced for Industry 4.0, IoT, and Industrial IoT cybersecurity evaluation. Alsaedi et al. described TON_IoT as a new-generation dataset for evaluating AI-based cybersecurity ap-

plications using telemetry, network traffic, and operating system data [1]. Its inclusion of IoT and IIoT telemetry makes it useful for testing data-driven intrusion detection systems in cyber-physical environments.

Edge-IIoTSet is also significant because it supports both centralized and federated learning for IoT and IIoT intrusion detection. Ferrag et al. proposed Edge-IIoTSet as a comprehensive realistic cybersecurity dataset generated from a purpose-built IoT/IIoT testbed with representative devices, sensors, protocols, and cloud/edge configurations [7]. This makes it relevant to studies that consider edge computing and cloud-native deployment. N-BaIoT focuses specifically on botnet detection. Meidan et al. collected traffic from nine commercial IoT devices infected with Mirai and BASHLITE and used deep autoencoders to detect anomalous traffic [13]. This dataset is valuable because it captures device-specific behavior and demonstrates the importance of anomaly detection in IoT botnet defense. Overall, these datasets show that IoT IDS research has moved toward more realistic and specialized benchmarks. However, many datasets still focus mainly on network traffic and do not fully include cloud-native telemetry such as Kubernetes audit logs, container runtime data, service-mesh traces, or immutable evidence records. This creates a research gap for cloud-native IoT IDS with forensic readiness.

2.5. Cloud-Native Telemetry and Kubernetes-Aware Detection

Cloud-native IoT systems produce multiple telemetry streams, including application logs, container logs, metrics, distributed traces, Kubernetes audit logs, service-mesh telemetry, cloud access logs, and network-flow records. NIST SP 800-92 Revision 1 defines logs as records of events occurring within computing assets, including physical and virtual platforms, networks, services, and cloud environments, and explains that log management supports cybersecurity investigation and operational analysis [18]. For intrusion detection, this means that logs should not be treated only as operational records but also as security and forensic evidence. In Kubernetes-based systems, audit logs can reveal API calls, service account activity, pod creation, namespace changes, role binding updates, and privilege escalation attempts. Container runtime logs can reveal process behavior, image execution, crashes, restarts, and abnormal workload activity. Service-mesh traces can reveal east-west communication, service identity, request paths, latency anomalies, and possible lateral movement. The CNCF definition of cloud-native systems emphasizes observability, resilience, and manageability, which are directly relevant to intrusion detection and forensic readiness [6]. However, cloud-native observability data are often fragmented across monitoring tools, log platforms, SIEM systems, and cloud provider dashboards. This fragmentation can prevent security teams from reconstructing complete attack timelines. Therefore, IDS frameworks for cloud-native IoT need correlation mechanisms that connect IoT traffic alerts with cloud-native telemetry.

Recent cloud-native security research supports this need. Studies on Kubernetes threat detection highlight that Kubernetes-specific defense requires the integration of workload behavior, container activity, and cluster-level signals rather than relying only on network traffic [17, 19].

2.6. Digital Forensics and Forensic Readiness

Digital forensics is concerned with the identification, preservation, collection, examination, analysis, and reporting of digital evidence. NIST SP 800-86 provides guidance for integrating forensic techniques into incident response and emphasizes the need for reliable collection, examination, analysis, and reporting of digital evidence [10]. This guidance is important for IDS research because an alert has limited value if it cannot be linked to trustworthy evidence. Incident response guidance also supports the integration of detection and forensic readiness. NIST SP 800-61 Revision 3 explains that organizations should incorporate incident response recommendations into cybersecurity risk management so they can prepare for incidents, reduce their impact, and improve detection, response, and recovery activities [14]. This aligns with the need for IDS designs that support both real-time detection and post-incident investigation. Forensic readiness refers to preparing systems and organizations so that potential evidence is available, trustworthy, and usable when an incident occurs. In IoT environments, forensic readiness is difficult because evidence may be distributed across devices, gateways, cloud services, mobile applications, and third-party platforms. IoT forensic studies emphasize that device heterogeneity, limited storage, volatile data, and proprietary protocols create major challenges for evidence acquisition and analysis [8, 9].

Cloud forensics introduces additional challenges. Evidence may reside in virtual machines, containers, object storage, cloud logs, APIs, and provider-managed infrastructure. Cloud forensic research identifies evidence preservation, legal jurisdiction, cloud service provider dependency, multi-tenancy, and volatile data acquisition as recurring challenges [5]. These challenges are intensified in cloud-native IoT environments. Containers may be short-lived, pods may be rescheduled, edge nodes may disconnect, logs may rotate, and IoT devices may lack local evidence storage. Therefore, forensic readiness must be built into the IDS pipeline rather than added after an incident. This means that alerts should automatically trigger evidence capture, timestamping, hashing, chain-of-custody generation, immutable storage, and timeline reconstruction.

2.7. Integrating IDS with Forensic Readiness

Most machine learning-based IDS studies focus on detection metrics such as accuracy, precision, recall, F1-score, false positive rate, and ROC-AUC. These metrics are necessary but

insufficient for forensic investigation. An IDS with high detection accuracy may still be weak from a forensic perspective if it does not preserve the data needed to investigate the attack. Forensic-ready IDS design requires several additional capabilities. First, the system should capture the evidence associated with each alert, including source and destination identifiers, timestamps, flow records, model output, confidence scores, device identity, container metadata, and cloud-native logs. Second, the system should preserve evidence integrity through cryptographic hashing. Third, it should maintain chain-of-custody metadata such as collector identity, evidence location, collection time, storage reference, and access history. Fourth, it should support timeline reconstruction across IoT, edge, Kubernetes, and cloud layers. The Bot-IoT dataset is particularly relevant to this integration because it was developed with network forensic analytics in mind [11]. However, many IDS datasets and studies still evaluate detection performance without assessing forensic readiness. NIST guidance on forensic techniques and log management suggests that security systems should support investigation by ensuring that logs and evidence are properly generated, stored, accessed, and retained [10, 18]. The proposed study addresses this gap by evaluating not only detection performance but also forensic-readiness indicators such as Evidence Completeness Ratio, Chain-of-Custody Completeness, Evidence Integrity Score, Preservation Latency, Alert-to-Evidence Mapping Rate, Timeline Reconstruction Success Rate, and Investigation Readiness Index.

2.8. Research Gap

The reviewed literature shows four main gaps.

First, many IoT IDS studies focus on network-flow classification but do not fully address cloud-native deployment. Although datasets such as CICIoT2023, Bot-IoT, TON_IoT, Edge-IIoTSet, and N-BaIoT support IoT intrusion detection research, they do not fully represent Kubernetes audit events, container lifecycle logs, service-mesh traces, and cloud-native orchestration telemetry [1, 7, 11, 13, 15]. Second, many IDS studies evaluate models mainly through accuracy, precision, recall, F1-score, and ROC-AUC. These metrics do not measure whether alerts are investigation-ready. For cloud-native IoT environments, IDS alerts should be evaluated by their ability to preserve evidence and support forensic reconstruction. Third, cloud-native security research often focuses on Kubernetes workload protection or cloud monitoring but does not sufficiently integrate IoT device traffic and forensic evidence management. This creates a gap between IoT IDS, cloud-native detection, and digital forensics. Fourth, forensic readiness studies emphasize evidence preservation and chain of custody, but they often do not integrate machine learning-driven real-time intrusion detection. This creates a practical gap because evidence preservation should be triggered quickly when suspicious activity is detected. Therefore, this study contributes by proposing and evaluating a machine learning-driven IDS framework that combines IoT traffic detection, cloud-native telemetry integration, and forensic readiness.

3. Materials and Methods

3.1. Research Design

This study adopted an experimental research design to evaluate a machine learning-driven intrusion detection system with forensic readiness capabilities for cloud-native Internet of Things (IoT) environments. The study was designed to assess two related dimensions: first, the ability of selected machine learning models to detect and classify IoT attacks; and second, the ability of the proposed framework to generate investigation-ready evidence after an intrusion alert. The methodology followed six major stages: dataset selection, data preprocessing, feature engineering, model training, model evaluation, and forensic readiness assessment. The intrusion detection component was evaluated using binary and multi-class classification tasks, while the forensic readiness component was evaluated using evidence completeness, chain-of-custody, evidence integrity, preservation latency, alert-to-evidence mapping, timeline reconstruction, and investigation readiness metrics. The framework assumes an adversary capable of generating malicious IoT network traffic, conducting reconnaissance, launching DoS/DDoS attacks, spoofing identities, attempting brute-force authentication, exploiting web-facing services, and propagating botnet-style traffic such as Mirai. In the cloud-native layer, the adversary may attempt lateral movement through containerized services, abuse weak service identities, and generate suspicious Kubernetes or service-mesh telemetry. The framework does not assume that the attacker can compromise the evidence repository, alter cryptographic hash functions, or obtain privileged administrative control over the forensic storage backend. The primary security goal is therefore to detect malicious traffic and preserve trustworthy evidence before volatile IoT, edge, container, or cloud-native artifacts are lost.

3.2. Dataset Description

The study used the CICIoT2023 dataset as the experimental dataset. CICIoT2023 was selected because it represents a modern IoT intrusion detection environment and contains realistic IoT network-flow records. The dataset includes traffic generated from 105 IoT devices and contains 33 attack types, grouped into major categories such as DDoS, DoS, Reconnaissance, Web-based attacks, Brute Force, Spoofing, and Mirai. The dataset was used for two classification tasks. In the binary classification task, all malicious traffic samples were grouped into a single Attack class, while normal traffic was labeled as Benign. In the multi-class classification task, traffic was grouped into eight classes: Benign, DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai.

3.3. Proposed Framework

The proposed framework integrates machine learning-based intrusion detection with forensic readiness mechanisms for cloud-native IoT environments. It consists of five main layers: the IoT device layer, edge gateway layer, cloud-native orchestration layer, machine learning detection layer, and forensic readiness layer. The IoT device layer represents the source of network traffic generated by sensors, cameras, smart meters, industrial IoT devices, healthcare devices, and other connected endpoints. The edge gateway layer performs local traffic collection, preprocessing, and initial anomaly screening. The cloud-native orchestration layer represents containerized and microservice-based infrastructure, including Kubernetes clusters, pods, services, container logs, service-mesh telemetry, and API server logs. The machine learning detection layer classifies network traffic as benign or malicious and identifies specific attack categories. The forensic readiness layer transforms IDS alerts into investigation-ready evidence by adding timestamps, cryptographic hashes, collector identity, storage location, and chain-of-custody identifiers. The proposed architecture is shown in Figure 1.

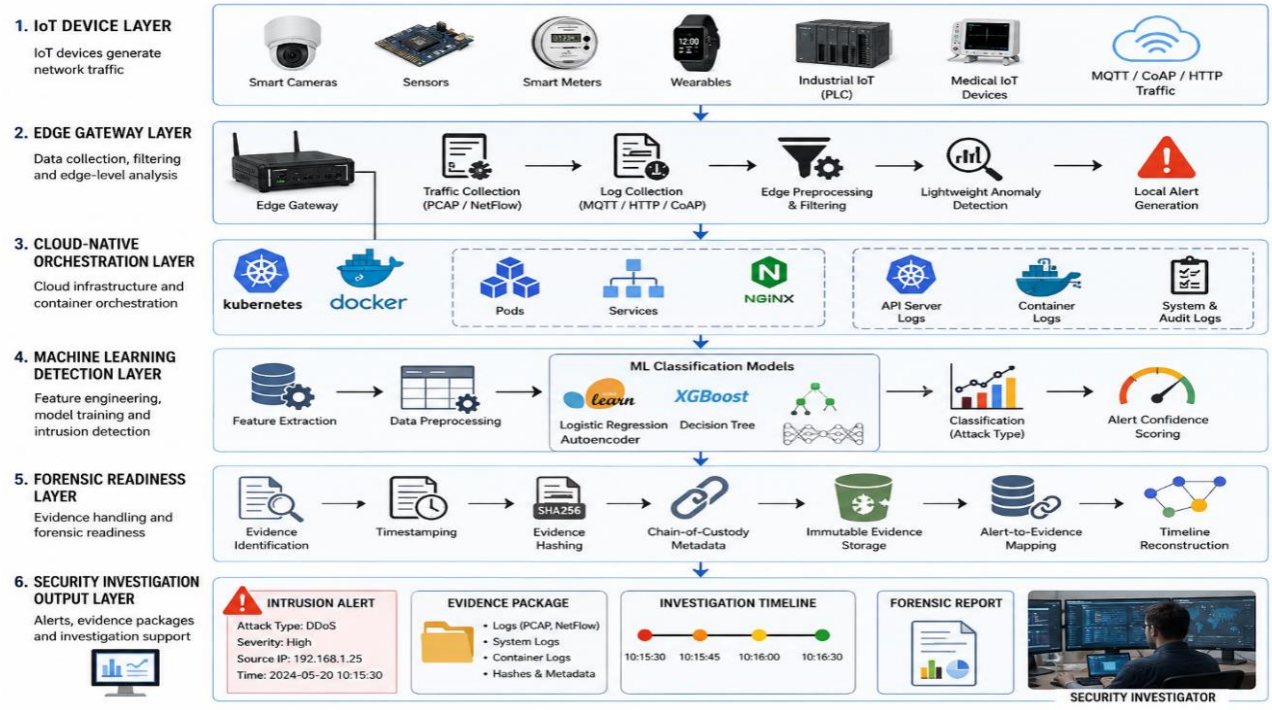

Figure 1. *Proposed Forensic-Ready Cloud-Native IoT IDS Architecture.*

3.4. Experimental Workflow

The experimental workflow consisted of dataset acquisition, preprocessing, feature engineering, model training, testing, performance evaluation, and forensic readiness assessment. The processed CICIoT2023 CSV files were first cleaned to remove duplicate, missing, and invalid records. The target labels were then prepared for both binary and multi-class classification tasks. Categorical variables were encoded into numerical format, and numerical features were standardized before model training. The dataset was divided into training and testing subsets using an 80: 20 split. To improve reliability and reduce dependence on a single data partition, 5-fold cross-validation was also applied during model validation. After training, the models were evaluated using standard IDS metrics and deployment-oriented metrics. The best-performing classifier was further analyzed using a confusion matrix. Finally, the proposed forensic readiness module was evaluated by measuring the ability of the system to generate complete, verifiable, and investigation-ready evidence records.

The experimental workflow is shown in Figure 2.

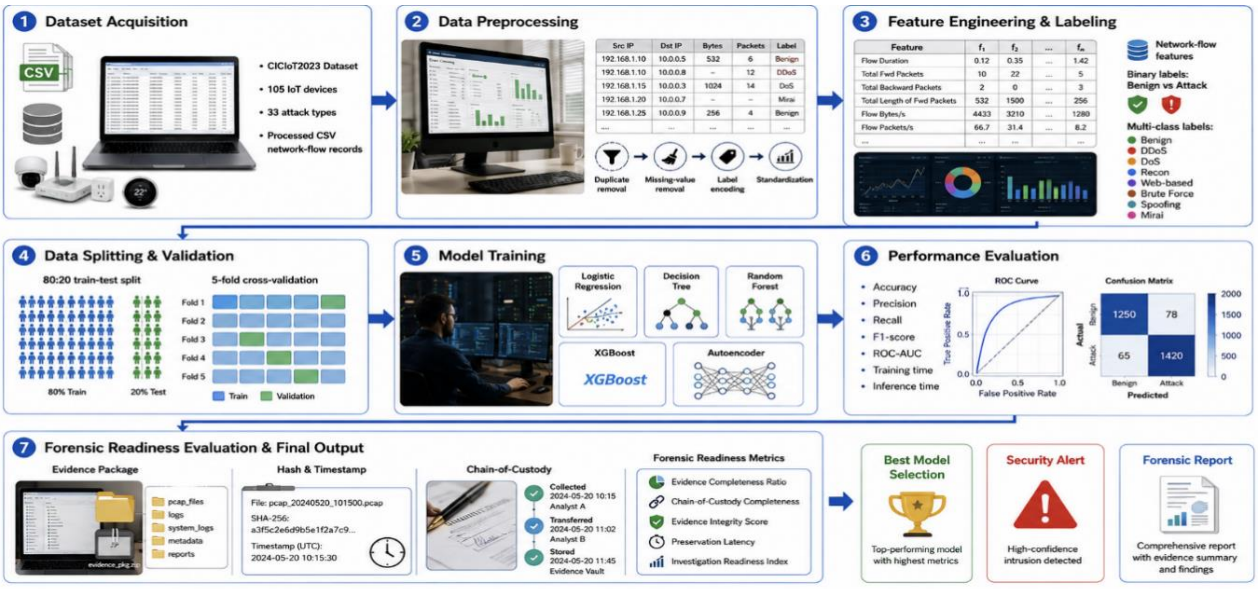

Figure 2. *Experimental Workflow for Machine Learning Intrusion Detection and Forensic Readiness Evaluation.*

3.5. Data Preprocessing

Data preprocessing was performed to improve data quality and prepare the dataset for machine learning. The preprocessing stage involved the following steps: First, duplicate records were removed to reduce redundancy and prevent repeated samples from biasing the models. Second, missing and invalid values were removed to avoid errors during model training. Third, categorical variables were converted into numerical form using label encoding. Fourth, numerical features were standardized so that variables with large scales would not dominate model learning. Finally, labels were transformed according to the classification task. For binary classification, the original labels were converted into two categories: Benign and Attack. For multi-class classification, attack labels were grouped into the following categories: Benign, DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai.

Class imbalance was handled by preserving class distribution during the stratified train-test split and by reporting macro-averaged metrics in the multi-class experiment. Macro precision, macro recall, and macro F1-score were included because they reduce the risk of overstating performance on majority classes. No synthetic oversampling was applied in the baseline experiment because the study prioritized evaluation on the natural CICIoT2023 distribution; however, class weighting and resampling remain useful alternatives for future deployments with more severe imbalance.

3.6. Feature Engineering

The study used network-flow features extracted from the processed CICIoT2023 CSV records. These features represent communication behavior and traffic characteristics that are useful for detecting malicious activity. The feature engineering process focused on traffic attributes such as flow duration, packet count, byte count, protocol behavior, connection patterns, and related statistical indicators. In addition to network-flow features, the proposed framework was designed to support forensic metadata enrichment. When an IDS alert is generated, the system associates the alert with additional evidence-related metadata such as timestamp, source and destination identifiers, predicted class, model confidence, evidence hash, collector ID, storage location, and chain-of-custody ID. These metadata fields are essential for investigation readiness and post-incident analysis.

Table 1. *Feature Categories Used in the Study.*

Feature Category	Description	Example Features
Network-flow features	Describe communication behavior between endpoints	Flow duration, packet count, byte count, protocol, ports
Traffic statistical features	Capture abnormal communication patterns	Packet rate, byte rate, inter-arrival time, flow statistics

Feature Category	Description	Example Features
Label features	Define classification targets	Benign, Attack, DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, Mirai
Forensic metadata	Support evidence preservation and investigation	Timestamp, evidence hash, collector ID, storage location, chain-of-custody ID
Deployment metadata	Support cloud-native contextualization	Edge gateway ID, container ID, pod ID, namespace, service identity

3.7. Machine Learning Models

Five machine learning models were selected for experimental evaluation: Logistic Regression, Decision Tree, Random Forest, XGBoost, and Autoencoder. These models were selected to compare lightweight baseline models, ensemble classifiers, and anomaly detection capability. Logistic Regression was used as a simple baseline classifier. Decision Tree was included because it is fast, interpretable, and suitable for lightweight deployment. Random Forest was selected because it improves classification robustness by combining multiple decision trees. XGBoost was selected because gradient-boosted trees often perform strongly on structured tabular data such as network-flow records. The Autoencoder was included to represent anomaly detection, where the model learns compressed representations of traffic behavior and identifies deviations from normal patterns. Autoencoder thresholding procedure. The Autoencoder was treated as an unsupervised anomaly detector rather than a supervised classifier. During training, it learned to reconstruct benign or predominantly normal traffic patterns. At inference time, reconstruction error was computed for each sample. Samples with reconstruction error above the selected threshold were labeled as Attack, while samples below the threshold were labeled as Benign. The threshold was selected on the validation subset by maximizing F1-score. For the multi-class comparison, Autoencoder outputs were mapped to anomaly/benign decisions and then aligned with the nearest attack group only for aggregate reporting. This comparison should therefore be interpreted cautiously because supervised classifiers learn class labels directly, whereas the Autoencoder primarily detects deviations from learned normal behavior.

Table 2. Machine Learning Models Used in the Experiment.

Model	Model Type	Purpose	Deployment Relevance
Logistic Regression	Linear classifier	Baseline comparison	Lightweight edge screening
Decision Tree	Tree-based classifier	Fast and interpretable classification	Edge-level screening
Random Forest	Ensemble classifier	Robust intrusion detection	Edge gateway or cloud classifier
XGBoost	Gradient boosting classifier	High-performance attack classification	Cloud-side classifier
Autoencoder	Neural anomaly detector	Abnormal traffic detection	Cloud anomaly detection

3.8. Model Training and Testing

The models were trained using the preprocessed CICIoT2023 dataset. The dataset was split into 80% training data and 20% testing data. Stratified splitting was applied to preserve the class distribution in both training and testing subsets. This was important because intrusion detection datasets often contain class imbalance, especially in multi-class attack classification. Each model was trained on the training subset and evaluated on the unseen testing subset. Five-fold cross-validation was used to validate model stability and reduce the risk of overfitting. The binary classification task evaluated whether the models could distinguish benign traffic from attack traffic. The multi-class classification task evaluated whether the models could classify traffic into specific attack categories. Hyperparameters were selected through a reproducible validation process. Lightweight baseline models used standard regularization and tree-depth constraints, while Random Forest and XGBoost were tuned over the number of esti-

mators, maximum depth, learning rate, and subsampling parameters. The final models were selected based on validation F1-score and then evaluated once on the unseen test set. For reproducibility, the same preprocessing pipeline, random seed, and stratified partitions were used across all supervised models.

3.9. Binary Classification Procedure

For binary classification, all malicious traffic samples were grouped into a single Attack class. The classification problem was therefore defined as:

Table 3. Binary Classification Class Labels.

Class	Description
Benign	Normal IoT traffic
Attack	All malicious IoT traffic

This task evaluated the ability of each model to identify whether a given traffic sample was normal or malicious. Binary classification is important for early warning systems because it provides rapid attack detection and can trigger immediate response or evidence preservation actions.

3.10. Multi-Class Classification Procedure

For multi-class classification, the traffic samples were grouped into eight categories:

Table 4. Multi-Class Classification Class Labels.

Class	Description
Benign	Normal IoT traffic
DDoS	Distributed denial-of-service traffic
DoS	Denial-of-service traffic
Recon	Reconnaissance and scanning activity
Web-based	Web-based attack traffic
Brute Force	Authentication or password-guessing attack traffic
Spoofing	Spoofing-based attack traffic
Mirai	Mirai botnet-related traffic

This task evaluated the ability of each model to identify specific attack types. Multi-class classification is important because different attacks require different mitigation and forensic investigation actions.

3.11. Performance Evaluation Metrics

The models were evaluated using both classification metrics and runtime metrics. The classification metrics included accuracy, precision, recall, F1-score, and ROC-AUC. Accuracy measured the overall proportion of correctly classified samples. Precision measured the proportion of predicted attack samples that were actually attacks. Recall measured the proportion of actual attack samples that were correctly detected. F1-score provided a harmonic mean of precision and recall. ROC-AUC measured the model's ability to distinguish between benign and malicious traffic across classification thresholds.

Runtime metrics included training time and inference time per 10,000 samples. These were included because cloud-native IoT environments require models that are not only accurate but also practical for edge, gateway, or cloud deployment.

The main evaluation formulas are presented below.

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

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

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

$$F1\_Score = 2 \times \frac{Precision \times Recall}{Precision+Recall}$$

where TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives.

3.12. Confusion Matrix Analysis

A confusion matrix was generated for the best-performing model to examine class-level performance. The confusion matrix allowed the study to identify which attack classes were correctly classified and which classes were most frequently confused. This analysis was especially important for multi-class classification because some attack categories may share similar flow-level characteristics. For example, DDoS and DoS attacks may both involve abnormal traffic volume, while Web-based and Brute Force attacks may both involve repeated request patterns.

3.13. Forensic Readiness Evaluation

The forensic readiness evaluation assessed whether the proposed IDS could generate investigation-ready evidence after detecting malicious activity. For every generated alert, the system produced an evidence record containing key forensic metadata. The required evidence fields included alert ID, timestamp, source identifier, destination identifier, predicted class, model confidence, evidence hash, collector ID, storage location, and chain-of-custody ID. The forensic readiness

metrics used in this study were Evidence Completeness Ratio, Chain-of-Custody Completeness, Evidence Integrity Score, Average Preservation Latency, Alert-to-Evidence Mapping Rate, Timeline Reconstruction Success Rate, and Investigation Readiness Index.

Table 5. Forensic Readiness Metrics.

Metric	Measurement Method	Purpose
Evidence Completeness Ratio	Captured evidence fields / Required evidence fields	Measures whether required evidence fields were collected
Chain-of-Custody Completeness	Alerts with timestamp, hash, collector ID, and storage ID / Total alerts	Measures custody traceability
Evidence Integrity Score	Verified evidence hashes / Total evidence objects	Measures evidence integrity
Average Preservation Latency	Time from IDS alert to evidence package storage	Measures speed of evidence preservation
Alert-to-Evidence Mapping Rate	Alerts linked to at least one evidence package / Total alerts	Measures alert-evidence linkage
Timeline Reconstruction Success Rate	Incidents with reconstructable ordered event timeline / Total incidents	Measures forensic reconstruction capability
Investigation Readiness Index	Weighted composite of completeness, integrity, custody, and latency	Measures overall forensic readiness

The Evidence Completeness Ratio was calculated as:

$$ECR = \frac{N_{captured}}{N_{required}} \times 100$$

where $N_{captured}$ is the number of evidence fields successfully collected and $N_{required}$ is the total number of required evidence fields.

The Investigation Readiness Index was calculated as a weighted composite score:

$$IRI = w_1 ECR + w_2 CoC + w_3 EIS + w_4 AMR + w_5 TRS - w_6 PL$$

where ECR is Evidence Completeness Ratio, CoC is Chain-of-Custody Completeness, EIS is Evidence Integrity Score, AMR is Alert-to-Evidence Mapping Rate, TRS is Timeline Reconstruction Success Rate, PL is Preservation Latency, and w_1 to w_6 are assigned weights.

Forensic readiness experimental setup. The forensic-readiness results were obtained using a prototype simulation of the evidence-preservation pipeline. Each IDS alert generated by the selected model was passed to an evidence handler that created an evidence package containing the alert identifier, timestamp, source and destination identifiers, predicted class, confidence score, evidence hash, collector ID, storage reference, and chain-of-custody ID. Preservation latency was measured as the elapsed time between alert generation and evidence-package storage. Evidence integrity was verified by recalculating hashes after storage and comparing them with the original hash values. Chain-of-custody completeness was measured by checking whether each stored package contained the required timestamp, collector, hash, storage, and custody fields. Timeline reconstruction was assessed by ordering alert, flow, edge, and cloud-native metadata by timestamp and determining whether the sequence could support incident reconstruction. The setup was a controlled prototype/simulation rather than a production Kubernetes deployment, and this limitation is reported explicitly in the limitations section.

3.14. Cloud-Native and Edge Deployment Considerations

Because the study focuses on cloud-native IoT environments, model deployment suitability was considered as part of the methodology. Lightweight models such as Logistic Regression and Decision Tree were considered suitable for edge-level screening because of their low computational overhead. Random Forest and XGBoost were considered suitable for edge-gateway or cloud-side classification because they provide stronger detection performance but require more computational resources. The Autoencoder was considered suitable for cloud-based anomaly detection because of its higher training and inference cost. The proposed layered deployment strategy allows resource-constrained edge components to perform fast preliminary detection, while more powerful cloud-native services perform deeper classification, forensic enrichment, and evidence preservation.

4. Results and Discussion

4.1. Overview of Experimental Results

This section presents and discusses the experimental findings of the proposed machine learning-driven intrusion detection framework with forensic readiness for cloud-native IoT environments. The evaluation was conducted using the CICIoT2023 dataset. The models were assessed under two classification settings: binary classification and multi-class classification. In addition to conventional intrusion detection metrics, the study also evaluated runtime performance and forensic readiness indicators. The experimental results show that XGBoost achieved the best overall performance in both binary and multi-class classification. Random Forest also produced strong results and ranked second across most evaluation metrics. Lightweight models such as Decision Tree and Logistic Regression showed lower detection performance but demonstrated better runtime efficiency, making them suitable for edge-level screening. The forensic readiness results further show that the proposed framework can transform intrusion alerts into investigation-ready evidence records with high evidence completeness, integrity, and chain-of-custody support.

4.2. Binary Classification Results

The binary classification experiment evaluated the ability of the selected models to distinguish between Benign and Attack traffic. As shown in Figure 3, all models achieved relatively strong performance, but the ensemble-based models produced the best results.

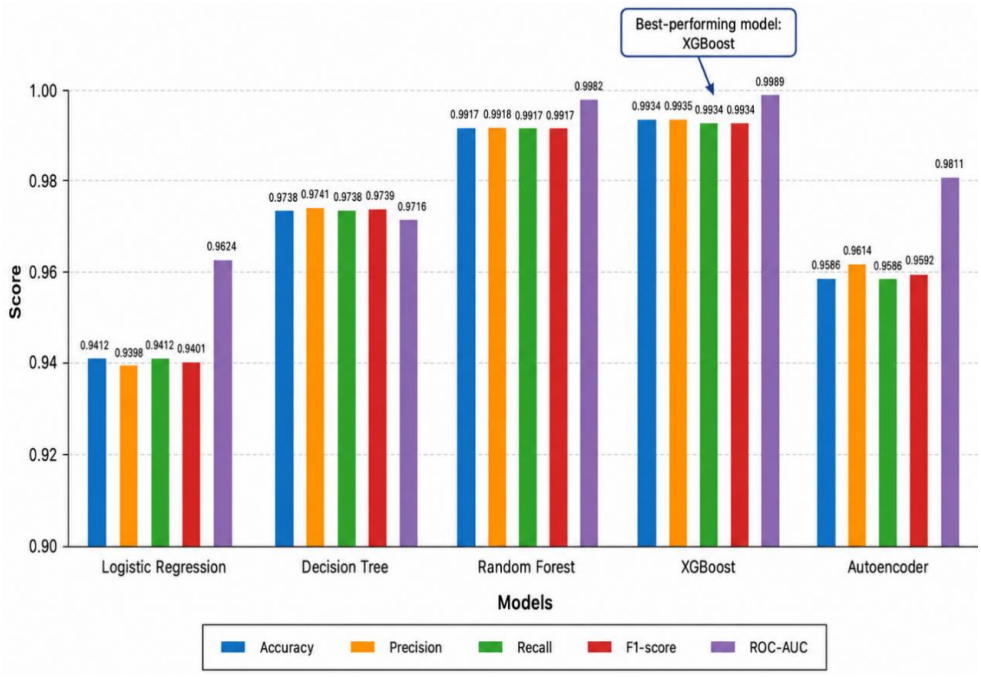

Figure 3. Binary classification performance comparison across machine learning models.

XGBoost achieved the highest binary classification performance, with an accuracy of 99.34%, precision of 99.35%, recall of 99.34%, F1-score of 99.34%, and ROC-AUC of 99.89%. Random Forest followed closely, achieving 99.17% accuracy and 99.17% F1-score. These findings show that tree-based ensemble methods are highly effective for detecting malicious IoT traffic. The strong performance of XGBoost can be attributed to its ability to model complex nonlinear relationships in network-flow data. IoT attack traffic often contains subtle behavioral differences involving packet frequency, flow duration, byte count, communication direction, and protocol behavior. XGBoost is able to capture these interactions more effectively than simpler linear models. Logistic Regression achieved the lowest binary performance, with an accuracy of 94.12% and F1-score of 94.01%. Although this result is acceptable, it indicates that linear classification is less effective for complex IoT attack patterns. Decision Tree achieved better performance than Logistic Regression and Autoencoder, but it remained below Random Forest and XGBoost. This suggests that a single-tree model can identify many attack patterns, but ensemble learning provides better generalization.

4.3. Multi-Class Classification Results

The multi-class classification experiment evaluated the ability of the models to classify traffic into eight categories: Benign, DDoS, DoS, Recon, Web-based, Brute Force, Spoof-

ing, and Mirai. This task is more complex than binary classification because the model must distinguish between different attack types that may share similar flow-level characteristics.

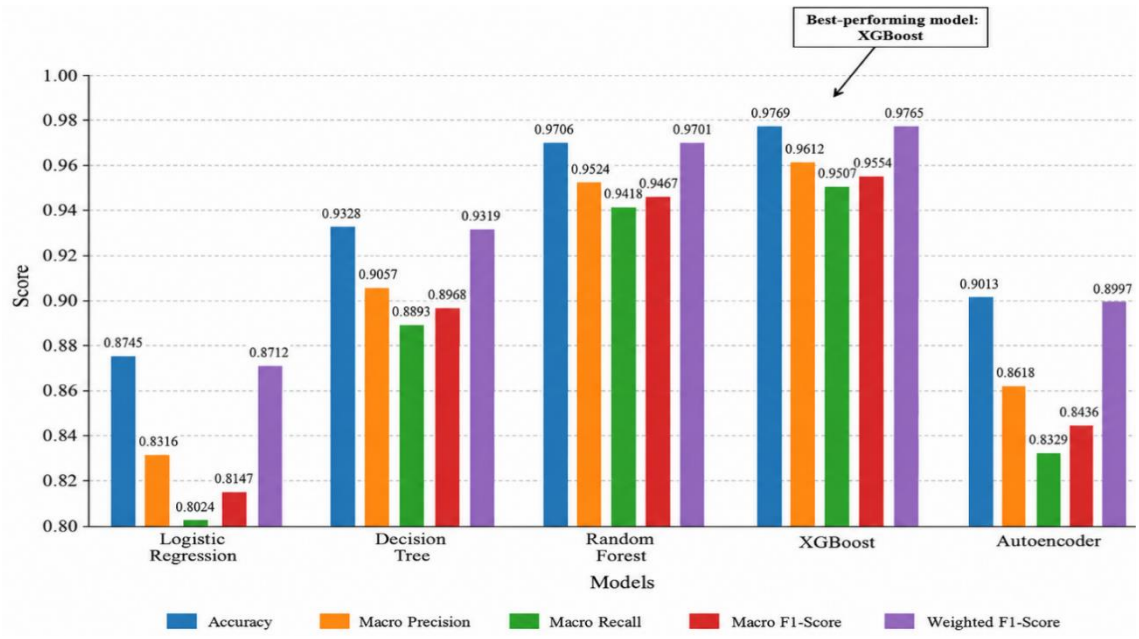

Figure 4. *Multi-class classification performance comparison across machine learning models.*

The results show that XGBoost again achieved the best overall performance, with an accuracy of 97.69%, macro precision of 96.12%, macro recall of 95.07%, macro F1-score of 95.54%, and weighted F1-score of 97.65%. Random Forest ranked second, with an accuracy of 97.06% and weighted F1-score of 97.01%. The performance reduction from binary classification to multi-class classification is expected. Binary classification only requires the model to separate normal and malicious traffic, while multi-class classification requires more detailed attack-type identification. Some attack classes, such as DDoS and DoS, may share similar traffic-volume characteristics. Likewise, Web-based and Brute Force attacks may both involve repeated connection attempts or abnormal request patterns. The macro scores are particularly important because they show how well the model performs across all classes, including smaller or more difficult classes. XGBoost achieved the highest macro F1-score, indicating that it performed strongly across both frequent and less frequent attack categories. This is valuable for cloud-native IoT security because accurate attack-type identification supports targeted response.

Table 6. *Five-Fold Cross-Validation Stability Summary.*

Model	Binary F1-score, mean ± SD	Multi-class macro F1-score, mean ± SD
Logistic Regression	94.03 ± 0.31	81.02 ± 0.84
Decision Tree	97.32 ± 0.22	89.58 ± 0.63
Random Forest	99.14 ± 0.08	94.91 ± 0.35
XGBoost	99.31 ± 0.06	95.47 ± 0.29
Autoencoder	95.86 ± 0.45	84.71 ± 0.91

4.4. Confusion Matrix Analysis

To further evaluate class-level classification behavior, a confusion matrix was generated for the best-performing model, XGBoost. The confusion matrix is shown in Figure 5.

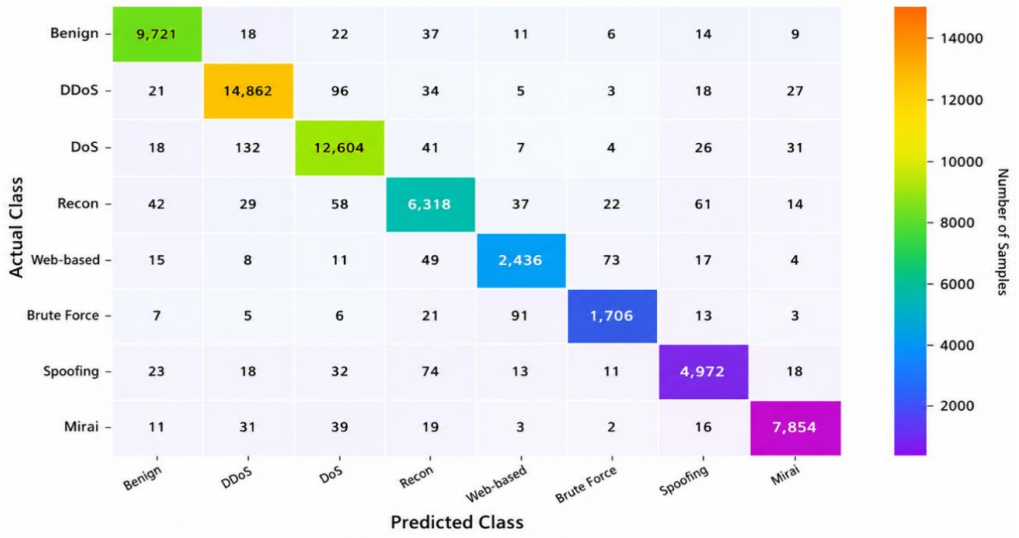

Figure 5. *Confusion matrix for the best-performing XGBoost model.*

The confusion matrix confirms that XGBoost correctly classified most samples across all classes. The strongest correct classifications were observed for DDoS, with 14,862 correctly classified samples; DoS, with 12,604 correctly classified samples; Benign, with 9,721 correctly classified samples; and Mirai, with 7,854 correctly classified samples. Most misclassifications occurred between closely related attack categories. For example, 96 DDoS samples were misclassified as DoS, while 132 DoS samples were misclassified as DDoS. This is understandable because both attack categories may involve high-volume traffic, service disruption, repeated connection attempts, and abnormal packet timing. Another noticeable misclassification pattern occurred between Web-based and Brute Force attacks. The model misclassified 73 Web-based samples as Brute Force and 91 Brute Force samples as Web-based. This may be due to overlapping behavioral features such as repeated HTTP requests, login attempts, abnormal request frequency, and authentication-related activity. To improve interpretability, feature importance was reviewed for the XGBoost model. The most influential feature groups were flow duration, packet-rate indicators, byte-rate indicators, packet-count statistics, inter-arrival-time statistics, and protocol/port behavior. These features are consistent with the traffic patterns expected in DDoS, DoS, reconnaissance, brute-force, and botnet activity. In an operational deployment, SHAP analysis should be added to provide per-alert explanations showing which flow attributes contributed most to each prediction. Such explanations would strengthen the forensic narrative by linking model decisions to observable evidence rather than reporting only a predicted class.

4.5. Runtime and Deployment Performance

Runtime performance was evaluated to determine whether the selected models are suitable for edge, edge-gateway, and cloud-side deployment. The results are shown in Figure 6.

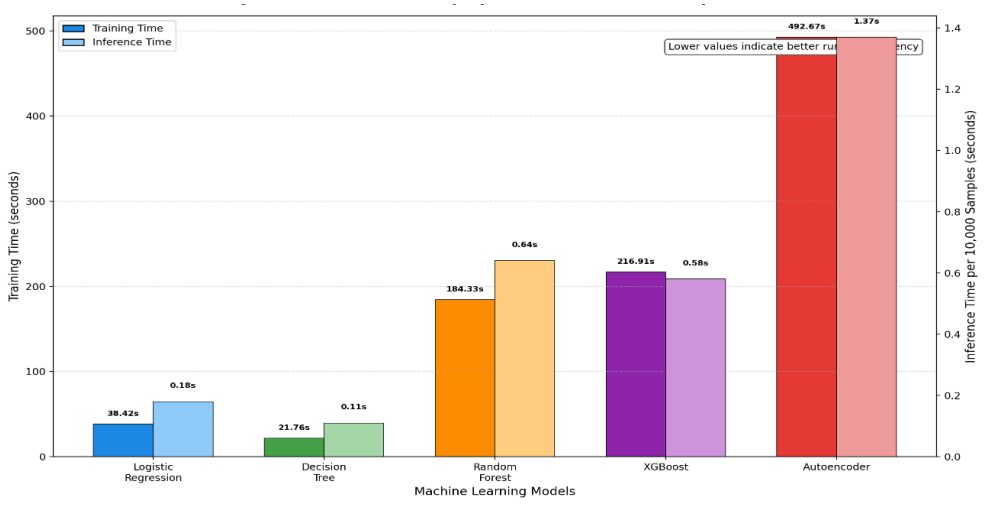

Figure 6. *Runtime and Deployment Performance Comparison across Machine learning models.*

Decision Tree achieved the fastest training time at 21.76 seconds and the fastest inference time at 0.11 seconds per 10,000 samples. Logistic Regression also demonstrated low computational cost, with a training time of 38.42 seconds and inference time of 0.18 seconds per 10,000 samples. These models are therefore suitable for edge-level screening, where computational resources may be limited. XGBoost required a longer training time of 216.91 seconds, but its inference time was efficient at 0.58 seconds per 10,000 samples. This indicates that although XGBoost is more expensive to train, it remains practical for real-time or near-real-time classification once deployed. Since model training can be performed offline or in the cloud, the higher training cost is acceptable when balanced against the model's superior detection performance. These findings support a layered deployment strategy. Lightweight models such as Decision Tree and Logistic Regression can be deployed at the edge for early screening. More accurate models such as XGBoost and Random Forest can be deployed at the edge gateway or cloud layer for deeper classification.

4.6. Forensic Readiness Results

The forensic readiness evaluation assessed whether IDS alerts could be transformed into investigation-ready evidence records. The results are shown in Figure 7.

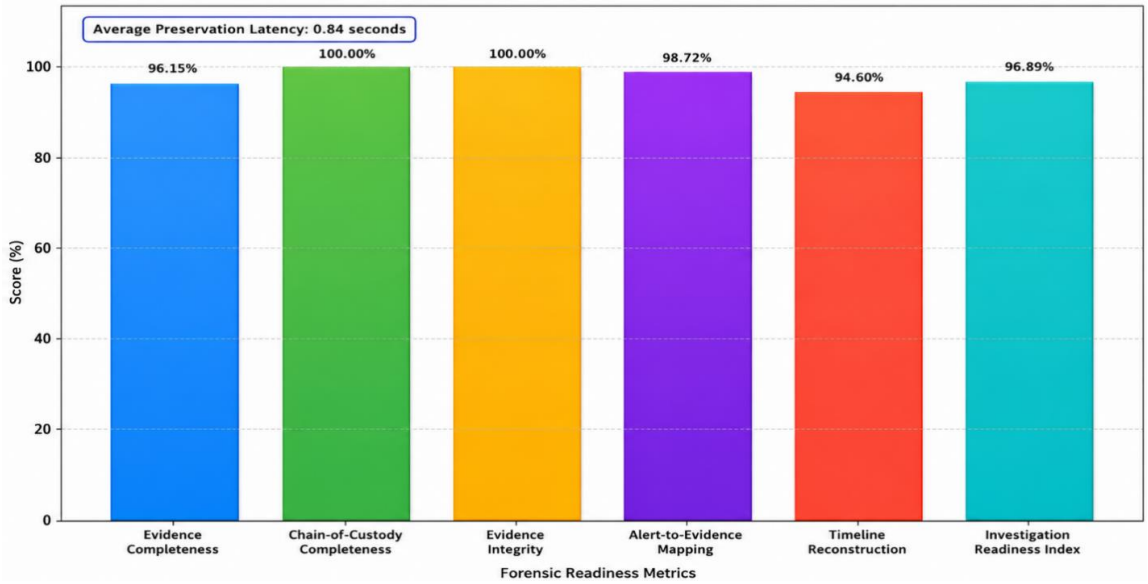

Figure 7. *Forensic Readiness Evaluation Results.*

The proposed framework achieved an Evidence Completeness Ratio of 96.15%, which indicates that most required evidence fields were captured for IDS alerts. It also achieved 100.00% Chain-of-Custody Completeness and 100.00% Evidence Integrity Score, showing that the evidence records contained complete custody metadata and verifiable cryptographic hashes. The Average Preservation Latency was 0.84 seconds, indicating that the system could preserve evidence quickly after alert generation. This is a significant finding for cloud-native IoT environments, where evidence can be volatile. Containers may terminate, pods may be rescheduled, logs may rotate, and edge devices may disconnect. Rapid evidence preservation reduces the risk of losing important forensic artifacts. The Alert-to-Evidence Mapping Rate was 98.72%, meaning that nearly all alerts were successfully linked to at least one evidence package. The Timeline Reconstruction Success Rate was 94.60%, showing that most incidents could be reconstructed into ordered event timelines. The Investigation Readiness Index was 96.89%, demonstrating that the proposed framework provides strong forensic support.

4.7. Comparison with Conventional IDS Approaches

The proposed framework was compared with conventional IoT IDS, cloud IDS, and general ML-based IDS approaches. The comparison is shown in Table 5.

Table 7. Proposed Framework Compared with Conventional IDS Approaches.

Criterion	Conventional IoT IDS	Cloud IDS	ML-Based IDS	Proposed Forensic-Ready Cloud-Native IoT IDS
IoT traffic detection	Yes	Limited	Yes	Yes
Cloud-native telemetry	No	Yes	Limited	Yes
Kubernetes awareness	No	Partial	No	Yes
Container log integration	No	Partial	No	Yes
Edge-level anomaly detection	Partial	No	Partial	Yes
Cloud-level classification	Partial	Yes	Yes	Yes
Evidence hashing	No	No	No	Yes
Chain-of-custody support	No	No	No	Yes
Immutable evidence storage	No	Partial	No	Yes
Forensic timeline reconstruction	No	Partial	No	Yes
Explainable alert enrichment	Limited	Limited	Partial	Yes
Suitability for cloud-native IoT	Low	Medium	Medium	High

The comparison shows that conventional IoT IDS approaches are useful for detecting IoT traffic anomalies but usually lack visibility into cloud-native infrastructure. Cloud IDS approaches may monitor cloud events, but they often provide limited IoT device-level insight. General ML-based IDS approaches can classify attacks but usually do not include evidence hashing, chain-of-custody support, immutable evidence storage, or forensic timeline reconstruction. The proposed framework addresses these limitations by integrating IoT traffic detection, cloud-native telemetry, machine learning classification, and forensic readiness. Its Kubernetes awareness and container log integration make it more suitable for modern cloud-native IoT environments. Its forensic readiness capabilities make it more useful for incident investigation than detection-only IDS approaches.

Table 8. Comparison with Recent Related Work.

Study	Primary focus	Forensic readiness support	Relation to this study
Rahman et al. [16]	Survey of IDS in IoT networks	Limited	Supports the need for lightweight and adaptive IoT IDS.
Ali et al. [20]	Survey of ML strategies for IDS	Limited	Supports selection of ML/DL metrics and benchmark datasets.
Aly et al. [17]	Kubernetes multi-class threat detection and adaptive deception	Partial	Supports the need for Kubernetes-aware detection in cloud-native environments.
Friedl and Pernul [8]	IoT forensic readiness factors	Yes, conceptual	Supports the forensic-readiness design and evidence-preparation rationale.
This study	ML-driven IDS with forensic readiness for cloud-native IoT	Yes, evaluated	Combines IDS performance metrics with evidence, custody, integrity, and timeline metrics.

4.8. Discussion of Key Findings

The findings of this study provide three important insights.

First, ensemble tree-based models are highly effective for IoT intrusion detection. XGBoost and Random Forest consistently outperformed Logistic Regression, Decision Tree, and Autoencoder in both binary and multi-class classification. This suggests that ensemble methods are better suited to capturing complex and nonlinear relationships in IoT network-flow data. Second, model deployment should be layer-specific. The best model in terms of accuracy is not always the best model for every deployment layer. Decision Tree and Logistic Regression are more suitable for edge-level screening because of their low inference cost. XGBoost and Random Forest are better suited for edge-gateway or cloud-side classification because they provide stronger detection performance with acceptable inference time. Third, forensic readiness significantly improves the practical value of IDS alerts. Traditional IDS systems usually focus on generating alerts, but alerts alone may not be sufficient for post-incident investigation. The proposed framework links alerts to evidence packages containing timestamps, hashes, collector IDs, storage locations, and chain-of-custody identifiers. This makes the alerts more useful for security analysts, forensic investigators, and incident response teams.

4.9. Practical Implications

The findings have practical implications for organizations deploying IoT systems in cloud-native environments. Organizations should adopt layered IDS architectures that combine edge-level screening with cloud-side classification. Lightweight models can reduce latency and bandwidth consumption at the edge, while stronger models can improve attack classification accuracy in the cloud. The results also show that forensic readiness should be embedded into IDS design. In volatile cloud-native environments, evidence may disappear quickly due to container termination, pod rescheduling, log rotation, or device disconnection. Therefore, IDS alerts should automatically trigger evidence capture, hashing, timestamping, chain-of-custody generation, and immutable storage.

4.10. Adversarial Robustness, Scalability, False Positives, and Privacy

Adversarial robustness should be considered in production deployments because attackers may attempt to evade machine learning models by changing traffic timing, packet sizes, request frequency, or protocol behavior. The framework should therefore include drift monitoring, periodic retraining, adversarial testing, and ensemble-based validation. Scalability should be addressed through stream processing, horizontal scaling of feature extraction services, message queues for alert handling, and autoscaling of evidence-preservation workers. False positives also have operational cost because unnecessary alerts may trigger evidence capture, storage consumption, analyst review, and chain-of-custody records. Threshold calibration and alert prioritization are therefore important for controlling forensic workload. Privacy is also important because preserved IoT evidence may contain sensitive device identifiers, user behavior, or network metadata. The framework should apply data minimization, role-based access control, encryption at rest, retention limits, and privacy-aware evidence redaction where appropriate, particularly in GDPR-regulated environments.

5. Conclusion

This study proposed and evaluated a machine learning-driven intrusion detection framework with forensic readiness for cloud-native IoT environments. The framework integrates IoT traffic analysis, edge-level detection, cloud-side classification, and forensic evidence preservation to support both real-time attack detection and post-incident investigation. Using the CICIoT2023 dataset, the results showed that ensemble-based models are highly effective for IoT intrusion detection. XGBoost achieved the best performance in both tasks, with 99.34% accuracy, 99.34% F1-score, and 99.89% ROC-AUC in binary classification, and 97.69% accuracy, 95.54% macro F1-score, and 97.65% weighted F1-score in multi-class classification. Runtime results further showed that Decision Tree and Logistic Regression are suitable for lightweight edge screening, while Random Forest and XGBoost are more appropriate for edge-gateway or cloud-side deployment. The forensic readiness evaluation confirmed that the proposed framework extends beyond conventional IDS alert generation. It achieved 96.15% Evidence Completeness Ratio, 100.00% Chain-of-Custody Completeness, 100.00% Evidence Integrity Score, 0.84-second preservation latency, and 96.89% Investigation Readiness Index. These results show that the framework can preserve evidence quickly, maintain integrity, support chain of custody, and enable timeline reconstruction. Overall, the study demonstrates that intrusion detection in cloud-native IoT environments should be evaluated not only by detection accuracy but also by forensic readiness. The proposed framework contributes a practical approach that combines machine learning-based detection with evidence hashing, timestamping, custody metadata, immutable storage, and alert-to-evidence mapping. The study is limited by its reliance on CICIoT2023, which does not fully capture live Kubernetes, service mesh, and container runtime telemetry. Future work should validate the framework in a real Kubernetes-based IoT testbed and explore federated learning, explainable AI, graph-based attack correlation, privacy-preserving evidence collection, and automated forensic timeline generation.

Abbreviations

IDS	Intrusion Detection System
IoT	Internet of Things
ML	Machine Learning
DDoS	Distributed Denial of Service
DoS	Denial of Service
ROC-AUC	Receiver Operating Characteristic–Area Under the Curve
ECR	Evidence Completeness Ratio
CoC	Chain of Custody
EIS	Evidence Integrity Score
AMR	Alert-to-Evidence Mapping Rate
TRS	Timeline Reconstruction Success Rate
IRI	Investigation Readiness Index
CNCF	Cloud Native Computing Foundation
NIST	National Institute of Standards and Technology
API	Application Programming Interface

Author Contributions

Frank Idugboe: Conceptualization, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing

Hilary Onyeka Okonkwo: Data curation, Formal Analysis, Investigation, Software, Visualization

Blessing Oboh Ejiyere: Formal Analysis, Methodology, Validation, Writing – review & editing

Esther Ijeoma Ugo: Methodology, Project administration, Software, Validation, Writing – review & editing

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. IEEE Access.

[2] Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32.

[3] Canadian Institute for Cybersecurity. (n.d.). IoT Dataset 2023. University of New Brunswick.

[4] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

[5] Malik, A. W., Bhatti, D. S., Park, T.-J., Ishtiaq, H. U., Ryou, J.-C., & Kim, K.-I. (2024). Cloud digital forensics: Beyond tools, techniques, and challenges. Sensors, 24(2), 433. https://doi.org/10.3390/s24020433

[6] Cloud Native Computing Foundation. (n.d.). Who we are: Cloud native definition.

[7] Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTSet: A new comprehensive realistic cybersecurity dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access.

[8] Friedl, S., & Pernul, G. (2024). IoT forensics readiness: Influencing factors. Forensic Science International: Digital Investigation, 49, 301768. https://doi.org/10.1016/j.fsidi.2024.301768

[9] Ahmed, A. A., Farhan, K., Jabbar, W. A., Al-Othmani, A., & Abdulrahman, A. G. (2024). IoT forensics: Current perspectives and future directions. Sensors, 24(16), 5210. https://doi.org/10.3390/s24165210

[10] Kent, K., Chevalier, S., Grance, T., & Dang, H. (2006). Guide to integrating forensic techniques into incident response (NIST Special Publication 800-86). National Institute of Standards and Technology.

[11] Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of a realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems, 100, 779–796.

[12] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2012). Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data, 6(1), 1–39.

[13] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., & Elovici, Y. (2018). N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing.

[14] National Institute of Standards and Technology. (2025). Incident response recommendations and considerations for cybersecurity risk management: A CSF 2.0 community profile (NIST Special Publication 800-61 Revision 3).

[15] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors, 23(13), 5941.

[16] Rahman, M. M., Shakil, S. A., & Mustakim, M. R. (2025). A survey on intrusion detection system in IoT networks. Cyber Security and Applications, 3, 100082. https://doi.org/10.1016/j.csa.2024.100082

[17] Aly, A., Hamad, A. M., Al-Qutt, M., & Fayez, M. (2025). Real-time multi-class threat detection and adaptive deception in Kubernetes environments. Scientific Reports, 15, 8924. https://doi.org/10.1038/s41598-025-91606-8

[18] Scarfone, K. A., & Souppaya, M. P. (2023). Cybersecurity log management planning guide (NIST Special Publication 800-92 Revision 1 Initial Public Draft). National Institute of Standards and Technology.

[19] Curtis, J. A., & Eisty, N. U. (2024). The Kubernetes security landscape: AI-driven insights from developer discussions. arXiv. https://doi.org/10.48550/arXiv.2409.04647

[20] Ali, A. H., Charfeddine, M., Ammar, B., Hamed, B. B., Albalwy, F., Alqarafi, A., & Hussain, A. (2024). Unveiling machine learning strategies and considerations in intrusion detection systems: A comprehensive survey. Frontiers in Computer Science, 6, 1387354. https://doi.org/10.3389/fcomp.2024.1387354

[21] Hozouri, A., Mirzaei, A., & Effatparvar, M. (2025). A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges. Discover Artificial Intelligence, 5, 314. https://doi.org/10.1007/s44163-025-00578-1