AI-Driven Intrusion Detection and Secure Coding for Cloud-Native Applications: A Layered Security Framework Review

Felix Ale[1], John A. Momoh[3], Emmanuel Segun Shokenu[1]*, Dawudu Ibrahim Shanzhi[1], Edet David Kokoette[1], Ahmed Adekunle Ajileye[1]

Corresponding Author: **Emmanuel Segun Shokenu**

Paper Number: 240332

Abstracts:

Cloud-native applications are increasingly vulnerable to sophisticated cyber threats due to their distributed architectures and dynamic environments. This study proposes a layered cybersecurity framework that integrates secure coding practices, cloud infrastructure protection, machine learningbased intrusion detection, and continuous monitoring for enhanced protection. Six machine learning modelsRandom Forest, Linear SVM, KNN, Gradient Boosting, Deep Learning, and Hybrid Stackingwere evaluated using accuracy, precision, recall, F1 score, false positive rate, and computational time. Results demonstrate that Random Forest and Hybrid Stacking achieved near-perfect performance, while all models effectively detected malicious activities with minimal false positives. Integrating these models within the layered framework addresses existing gaps in AI-driven security, including high false positive rates, limited adaptability to cloud-native environments, and the lack of coordination between detection and secure coding practices. The proposed approach provides a comprehensive, scalable, and practical solution for securing modern cloud-native applications, offering a foundation for adaptive and automated cyber security strategies.

Keywords: *Machine Learning, Intrusion Detection System (IDS), Cloud-Native Security, Secure Coding, Cybersecurity, Layered Security Framework*

Introduction

Cybersecurity has become a critical concern in modern software systems due to the rapid expansion of digital infrastructure, cloud computing, and interconnected applications. Organizations increasingly rely on distributed cloud services and microservice architectures to support scalable and flexible systems. However, these technologies also introduce new security challenges, including unauthorized access, data breaches, malware propagation, and advanced persistent threats. Recent studies emphasize that cyber attackers are continuously evolving their techniques, making

traditional rule-based security systems insufficient for protecting modern applications. Artificial intelligence (AI) has therefore emerged as a promising solution for strengthening cyber defense through intelligent threat detection and automated response mechanisms (Sarraf & Pal, 2026; Wang & Xie, 2025; Mollah, 2025).

Cloud-native applications, which rely heavily on containerization, microservices, and dynamic orchestration platforms, present additional vulnerabilities across multiple layers of the software stack. Weak application programming interfaces (APIs), insecure container configurations, and poorly implemented authentication mechanisms can expose critical systems to cyber threats. Researchers have highlighted the importance of integrating AI-driven intrusion detection systems (IDS) into cloud infrastructures to monitor network traffic and detect abnormal behaviors in real time (Correia, 2024; Olaoye, 2025; Anders, 2026). Furthermore, secure coding practices embedded within DevSecOps pipelines can significantly reduce vulnerabilities during software development and deployment stages (Singh, 2025; Sree et al., 2025; Brogi, 2023). The integration of AI-enabled analytics, automated monitoring, and predictive threat intelligence is increasingly recognized as a key approach for improving cyber resilience in cloud-native environments (Shonubi & Adelere, 2025; Sánchez, 2025; Prasad, 2024).

Despite the growing adoption of AI in cybersecurity, several research gaps remain. Many existing intrusion detection models suffer from high false positive rates and limited adaptability to complex cloud-native architectures. In addition, current security approaches often treat AI-based detection and secure coding practices as separate processes rather than integrated components of a comprehensive defense strategy. This lack of coordination reduces the overall effectiveness of security frameworks in distributed environments (Satyam et al., 2025; Zaka et al., 2025; Priya et al., 2025). Therefore, this study aims to address these challenges by proposing a layered cybersecurity framework that combines AI-driven intrusion detection mechanisms with secure coding principles for cloud-native applications. The objectives of this research are to review recent advances in AI-driven intrusion detection systems, analyze the role of secure coding practices in mitigating cyber threats, and propose a layered security framework that enhances protection for modern cloud-native software systems (Kashiv, 2025; Holmberg, 2025; Michael, 2025; Karthick, 2025; Segun et al., 2023; Samuel et al.; Ajayi et al., 2026a; Ajayi et al., 2026b).

Cloud-Native Applications

Cloud-native applications have become a dominant paradigm in modern software development due to their scalability, flexibility, and ability to

support distributed computing environments. These applications are typically built using microservices architectures, containerization technologies, and automated orchestration platforms such as Kubernetes, allowing organizations to rapidly deploy and manage services in dynamic cloud environments. The adoption of cloud-native systems enables efficient resource utilization and continuous integration and delivery (CI/CD), which significantly improves software agility and performance. However, the increased complexity and distributed nature of these architectures introduce new security concerns that require advanced protection mechanisms. According to Anders M. J. and A. J. Sánchez, cloud-native infrastructures integrate multiple services, APIs, and network components that operate across distributed environments, thereby expanding the potential attack surface and making traditional security solutions insufficient for modern cloud ecosystems. Furthermore, the integration of artificial intelligence and automation within cloud-native platforms has accelerated the evolution of intelligent cybersecurity frameworks designed to protect complex cloud infrastructures from emerging threats (Shonubi&Adelere, 2025; Sree et al., 2025; Prasad, 2024).

Cybersecurity Challenges in Cloud Environments

Despite the advantages of cloud-native architectures, they introduce significant cybersecurity challenges due to their distributed and highly dynamic nature. Microservices communicate through APIs and network protocols, which can expose vulnerabilities if not properly secured. Additionally, containerized environments often share system resources and kernels, increasing the risk of container escape attacks, unauthorized access, and misconfiguration vulnerabilities. According to Correia (2024) and Kashiv (2025), the decentralized architecture of cloud-native systems creates multiple entry points that can be exploited by attackers through network intrusion, privilege escalation, and distributed denial-of-service (DDoS) attacks. Moreover, cloud infrastructures handle massive volumes of data and traffic, making real-time monitoring and threat detection increasingly difficult using conventional security tools. Studies also highlight that attackers frequently target APIs, orchestration platforms, and container registries within cloud ecosystems, thereby increasing the risk of sophisticated cyberattacks (Sarraf & Pal, 2026; Singh, 2025). Consequently, modern cybersecurity strategies must integrate intelligent monitoring and adaptive defense mechanisms to ensure the protection of cloud-native services and infrastructure.

Importance of Machine Learning in Intrusion Detection

Machine learning has emerged as a powerful approach for enhancing intrusion detection systems due to its capability to analyze large volumes of network traffic and identify abnormal behavior patterns. Unlike traditional rule-based security systems, machine learning algorithms can learn from historical data and automatically adapt to new and evolving attack strategies. Several studies have demonstrated that supervised and unsupervised learning techniques such as Random Forest, Support Vector Machines, and Neural Networks can effectively detect anomalies in network traffic and classify malicious activities with high accuracy (Musleh et al., 2023; Amanoul et al., 2021). Additionally, machine learning models can process complex datasets containing network features such as packet size, protocol type, traffic flow, and connection duration to distinguish between legitimate and malicious behavior (Abdallah &Otoom, 2022). As highlighted by Wang and Xie (2025) and Zaka et al. (2025), integrating artificial intelligence with cybersecurity systems enables automated threat detection and predictive security analytics, making machine learning an essential component of modern intrusion detection frameworks.

4. Research Motivation

The increasing adoption of cloud-native computing has significantly expanded the digital attack surface, making traditional security mechanisms inadequate for protecting modern distributed systems. Conventional intrusion detection systems rely heavily on predefined signatures and static rule sets, which are ineffective against advanced persistent threats, zero-day attacks, and rapidly evolving cyber threats. Furthermore, the large volume of network data generated by cloud-native environments makes manual monitoring impractical and inefficient. These limitations highlight the urgent need for intelligent security mechanisms capable of detecting and responding to cyber threats in real time. According to Olaoye (2025) and Mollah (2025), integrating machine learning into cybersecurity frameworks enables proactive threat detection by identifying abnormal network behaviors and predicting potential attacks before they cause significant damage. Additionally, recent advancements in AI-driven cybersecurity research emphasize the need for combining machine learning-based intrusion detection with secure software development practices to establish a comprehensive defense strategy for cloud infrastructures (Priya et al., 2025; Satyam et al., 2025).

5. Contribution of the Study

This review study aims to analyze recent developments in machine learning-based intrusion detection and secure coding techniques for cloud-native

applications while proposing a layered security framework that integrates both approaches. The study provides a comprehensive overview of cloud-native security architectures, identifies major vulnerabilities in microservices and containerized environments, and examines existing machine learning algorithms used for intrusion detection. Additionally, it highlights the limitations of traditional intrusion detection mechanisms and discusses how artificial intelligence can enhance threat detection accuracy and automation. By synthesizing findings from recent cybersecurity research, this paper proposes a conceptual framework that combines secure coding practices with AI-driven intrusion detection to strengthen cloud-native security infrastructures. According to recent surveys on AI-based cybersecurity systems, combining intelligent threat detection with proactive software security practices significantly improves the resilience of modern cloud computing environments (Vanin et al., 2022; Hossain & Islam, 2023; Ahmad et al., 2021). The proposed framework therefore contributes to the advancement of cloud security by promoting a layered defense model capable of addressing emerging cyber threats.

Literature Review

Artificial Intelligence (AI) has significantly transformed intrusion detection systems by enabling automated detection of malicious activities in complex network environments. Traditional security mechanisms often rely on static rule-based approaches that struggle to detect evolving cyber threats in cloud-native infrastructures. Recent research highlights that AI-driven intrusion detection systems (IDS) can analyze large volumes of network traffic and identify abnormal behavior patterns with higher accuracy. Techniques such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and hybrid machine learning models have been widely adopted for cybersecurity analytics. Deep neural networks are capable of extracting complex hierarchical features from network data, making them suitable for detecting unknown attacks, while CNN models excel in feature extraction from high-dimensional datasets. LSTM models are particularly useful for analyzing sequential data such as network traffic logs and time-based attack patterns. Hybrid machine learning approaches combine multiple algorithms to improve detection accuracy and reduce false positives in intrusion detection systems (Correia, 2024; Sarraf & Pal, 2026; Wang & Xie, 2025; Olaoye, 2025; Priya et al., 2025).

Several benchmark datasets have been widely used to evaluate the performance of AI-based intrusion detection systems. The KDD Cup 99 dataset was one of the earliest datasets used for training and testing intrusion detection models, although it contains redundant records that

may bias machine learning models. To address these limitations, the NSL-KDD dataset was later introduced to improve data balance and remove duplicate records. More recent datasets such as CICIDS2017 and UNSW-NB15 provide more realistic representations of modern network traffic and include diverse cyberattack scenarios such as Distributed Denial of Service (DDoS), brute force attacks, infiltration attacks, and botnet activity. These datasets offer richer feature sets and realistic traffic patterns, which are important for evaluating deep learning algorithms used in modern IDS systems. However, their large size and complexity require significant computational resources and careful preprocessing before model training (Satyam et al., 2025; Mollah, 2025; Zaka et al., 2025; Kashiv, 2025; Karthick, 2025).

In addition to AI-driven detection mechanisms, secure coding practices play a crucial role in preventing vulnerabilities during the software development lifecycle. Security standards such as the Open Web Application Security Project Secure Coding Guidelines, International Organization for Standardization, and the National Institute of Standards and Technology provide structured approaches for implementing security controls in cloud-native applications. Key secure coding practices include input validation, strong authentication mechanisms, encryption of sensitive data, secure API design, and dependency vulnerability scanning. Failure to implement these practices can expose systems to serious cyber threats such as SQL injection, cross-site scripting (XSS), and privilege escalation attacks. Integrating secure coding practices with AI-driven intrusion detection systems therefore provides a comprehensive defense strategy for protecting modern cloud infrastructures and microservice-based applications (Singh, 2025; Sree et al., 2025; Brogi, 2023; Anders, 2026; Shonubi&Adelere, 2025; Prasad, 2024; Michael, 2025).

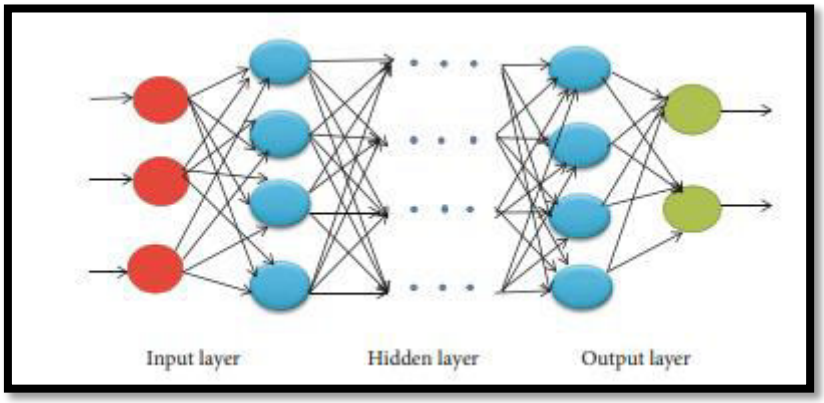

Figure 1: Deep Learning Neural Network Architecture for Intrusion Detection (Source: Sumathi et al., 2022)

The figure 1 illustrates the architecture of deep learning networks used for intrusion detection systems that analyze network traffic and detect

cyberattacks such as DDoS. Deep learning architectures have been widely applied for detecting cyberattacks in network environments. A typical deep neural network architecture used for intrusion detection includes multiple hidden layers that extract hierarchical features from network traffic data, improving the detection of malicious patterns. Such architectures have been used successfully for detecting distributed denial-of-service attacks and other network intrusions (Sumathi, Rajesh, & Lim, 2022).

Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) are critical cybersecurity tools designed to monitor network traffic and system activities in order to detect malicious behavior, unauthorized access, or policy violations within computer systems. IDS technologies analyze network packets, system logs, and application activities to identify potential security threats and alert administrators about suspicious activities. As cyberattacks continue to evolve in complexity and frequency, IDS solutions have become essential components of modern security infrastructures, particularly in cloud and distributed environments. Traditional security mechanisms such as firewalls provide perimeter protection; however, they are often insufficient for identifying sophisticated attacks that occur within internal networks. According to Abdallah and Otoom (2022) and Dina and Manivannan (2021), IDS solutions enhance network security by continuously monitoring system behavior and detecting abnormal patterns that may indicate malicious activities. Recent research has also highlighted the integration of artificial intelligence techniques into IDS frameworks to improve detection accuracy and reduce false positives in large-scale network environments (Vanin et al., 2022; Dini et al., 2023).

Types of Intrusion Detection Systems

Intrusion detection systems are generally categorized into two primary types based on the location of monitoring and the source of data used for detecting attacks: Host-Based Intrusion Detection Systems (HIDS) and Network-Based Intrusion Detection Systems (NIDS). These systems complement each other by providing different levels of visibility into system operations and network activities.

Host-Based Intrusion Detection Systems (HIDS)

Host-Based Intrusion Detection Systems (HIDS) operate by monitoring activities occurring within individual hosts or devices such as servers, workstations, or virtual machines. These systems analyze system logs, file integrity, user behavior, operating system processes, and application activities to detect suspicious behavior that may indicate an intrusion attempt. HIDS solutions are particularly effective in detecting insider threats, unauthorized file modifications, and privilege escalation attacks because they operate directly within the monitored system. According to

Ahmad et al. (2021), host-based detection systems provide deep visibility into system-level activities that cannot be easily observed by network-based monitoring tools. Additionally, HIDS can track changes in critical system files and detect malicious software or unauthorized configuration changes that compromise system integrity. However, researchers note that HIDS may require significant computational resources and can be difficult to manage in large-scale cloud environments where numerous hosts and containers are deployed (Kocher & Kumar, 2021).

Network-Based Intrusion Detection Systems (NIDS)

Network-Based Intrusion Detection Systems (NIDS) monitor network traffic flowing across communication channels to detect suspicious activities or potential cyberattacks. These systems are typically deployed at strategic locations within a network, such as gateways, routers, or switches, where they can analyze incoming and outgoing packets in real time. NIDS solutions inspect packet headers, communication protocols, traffic flow patterns, and network connections to identify anomalies or known attack signatures. According to Bertoli et al. (2021), NIDS systems are widely used in enterprise networks because they provide centralized monitoring capabilities and can detect large-scale attacks such as distributed denial-of-service (DDoS), port scanning, and network-based malware propagation. Furthermore, network-based monitoring is particularly useful in cloud and distributed environments where multiple systems communicate over shared network infrastructures. However, traditional NIDS may struggle with encrypted traffic and high-speed network environments, which has motivated the integration of machine learning techniques to enhance detection performance (Reddy & Reddy, 2025; Pinto et al., 2023).

Detection Techniques in Intrusion Detection Systems

Intrusion detection systems rely on different detection techniques to identify malicious activities in computer networks. The most common approaches include signature-based detection, anomaly-based detection, and hybrid detection techniques. Each approach has its advantages and limitations depending on the type of cyber threats being addressed and the environment in which the IDS is deployed.

Signature-Based Detection

Signature-based intrusion detection relies on predefined patterns or signatures that represent known cyberattack behaviors. These signatures are typically derived from previously identified malware, exploit patterns, or attack sequences stored in security databases. When incoming network traffic or system activity matches a known attack signature, the IDS generates an alert indicating a potential security breach. Signature-based detection methods are highly effective in identifying well-known threats and previously documented attack patterns. According to Agarwal et al. (2021),

this technique is widely used in traditional security systems because it provides high accuracy for detecting known attacks with relatively low computational overhead. However, a major limitation of signature-based detection is its inability to detect new or unknown attacks, commonly referred to as zero-day vulnerabilities. As cyber threats continue to evolve, researchers emphasize the need for more adaptive detection techniques capable of identifying novel attack behaviors (Hidayat et al., 2023).

Anomaly-Based Detection

Anomaly-based intrusion detection techniques identify attacks by detecting deviations from normal system or network behavior. These systems first establish a baseline model of legitimate network activity using historical data and then monitor real-time traffic to identify abnormal patterns that may indicate malicious activities. Machine learning algorithms are often used to build behavioral models that can recognize unusual traffic patterns, unauthorized access attempts, or abnormal system operations. According to Musleh et al. (2023) and Hossain and Islam (2023), anomaly-based IDS approaches are particularly effective in detecting unknown threats, advanced persistent threats, and zero-day attacks because they do not rely solely on predefined signatures. Instead, they analyzebehavioral patterns to identify suspicious activities that differ from normal operations. Despite these advantages, anomaly-based systems may generate higher false positive rates if the baseline behavior model is not properly trained or updated (Kocher & Kumar, 2021).

Hybrid Intrusion Detection Systems

Hybrid intrusion detection systems combine both signature-based and anomaly-based detection techniques to improve overall detection performance. By integrating the strengths of both approaches, hybrid IDS frameworks can effectively detect both known attack patterns and previously unseen cyber threats. For example, signature-based components provide accurate detection of well-documented attacks, while anomaly detection algorithms analyze network behavior to identify unusual activities that may indicate new threats. According to Liu et al. (2021) and Saif et al. (2022), hybrid IDS models often utilize machine learning techniques such as ensemble learning, clustering algorithms, and deep learning models to enhance detection accuracy and reduce false alarms. These systems are particularly beneficial in modern cloud and Internet of Things (IoT) environments where large volumes of network traffic and diverse attack patterns require adaptive and intelligent security mechanisms. Consequently, hybrid intrusion detection systems are increasingly being adopted in cybersecurity research as a robust solution for protecting complex network infrastructures.

Machine Learning Techniques for Intrusion Detection

Machine learning techniques have significantly improved the effectiveness of intrusion detection systems by enabling automated analysis of network traffic and system behavior. Traditional rule-based security systems depend heavily on predefined signatures and static rules, which limits their ability to detect new and evolving cyber threats. Machine learning approaches address this limitation by learning patterns from historical network data and identifying deviations that may indicate malicious activities. These techniques can analyze large-scale datasets containing network features such as packet size, protocol type, connection duration, and traffic flow behavior to accurately classify normal and malicious activities. According to Abdallah and Otoom (2022) and Ahmad et al. (2021), machine learning-based IDS models can significantly improve detection accuracy and adaptability compared to traditional security approaches. Furthermore, recent studies highlight that integrating artificial intelligence into cybersecurity systems allows for automated threat detection and predictive security analytics capable of identifying emerging cyber threats in real time (Vanin et al., 2022; Wang & Xie, 2025). As a result, machine learning has become a core component of modern intrusion detection frameworks deployed in cloud computing and distributed network environments.

Supervised Learning Models

Supervised learning models are widely used in intrusion detection systems because they rely on labeled datasets that contain both normal and malicious network activities. These models learn from training data by identifying patterns associated with different attack types and then apply this knowledge to classify new network traffic. Common supervised machine learning algorithms used in IDS include Decision Trees, Random Forest, Support Vector Machines (SVM), and Logistic Regression. These algorithms are capable of identifying complex relationships between network features and attack patterns, thereby improving the detection of malicious activities. According to Amanoul et al. (2021) and Agarwal et al. (2021), supervised learning models have demonstrated high accuracy in detecting various types of cyberattacks such as denial-of-service attacks, probing attacks, and unauthorized access attempts. Additionally, ensemble-based machine learning approaches, which combine multiple classifiers, have been shown to enhance detection performance by reducing classification errors and improving model robustness (Hossain & Islam, 2023). However, the performance of supervised learning models depends heavily on the availability of high-quality labeled datasets, which may be difficult to obtain in real-world cybersecurity environments.

Unsupervised Learning Models

Unsupervised learning models play an important role in intrusion detection systems, particularly for identifying unknown attacks and anomalies in network traffic. Unlike supervised learning techniques, unsupervised models do not require labeled datasets and instead identify patterns or clusters in data based on similarities between network features. Algorithms such as K-means clustering, Isolation Forest, and density-based clustering methods are commonly used to detect abnormal behavior in network traffic. These models establish a baseline of normal system activity and flag deviations that may represent potential intrusions. According to Dina and Manivannan (2021) and Pinto et al. (2023), unsupervised learning methods are highly effective in detecting zero-day attacks and emerging cyber threats because they analyzebehavioral anomalies rather than relying on predefined attack signatures. Furthermore, these techniques are particularly useful in cloud and IoT environments where new attack patterns frequently emerge and labeled datasets may not be available. However, researchers also note that unsupervised models may produce higher false positive rates because normal system behavior can vary significantly in complex network environments (Kocher & Kumar, 2021).

Deep Learning-Based Intrusion Detection Systems

Deep learning has emerged as an advanced approach for developing highly intelligent intrusion detection systems capable of analyzing complex and high-dimensional network data. Deep learning models such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks are capable of automatically extracting relevant features from raw network traffic without requiring extensive manual feature engineering. These models can identify subtle patterns and relationships in network behavior that may be difficult for traditional machine learning algorithms to detect. According to Ashiku and Dagli (2021) and Mendonca et al. (2022), deep learning-based IDS frameworks have demonstrated superior performance in detecting sophisticated cyberattacks within large-scale network environments. Additionally, deep learning techniques are particularly effective in analyzing sequential network data and time-dependent attack patterns, making them suitable for real-time intrusion detection in cloud computing and Internet of Things infrastructures. Despite these advantages, deep learning models often require large datasets and significant computational resources for training, which may limit their deployment in resource-constrained environments (Dini et al., 2023).

Comparative Analysis of Machine Learning Algorithms for IDS

Comparative studies of machine learning algorithms have shown that different models exhibit varying levels of performance depending on the

dataset characteristics, network environment, and attack types being detected. Supervised learning models such as Random Forest and Support Vector Machines typically provide high classification accuracy and strong generalization capabilities for known attack patterns. In contrast, unsupervised learning algorithms are more suitable for detecting unknown threats and behavioral anomalies within network traffic. Deep learning models, on the other hand, provide advanced pattern recognition capabilities that enable them to detect complex cyberattacks with improved accuracy. According to Hidayat et al. (2023) and Turukmane and Devendiran (2024), ensemble learning and hybrid models that combine multiple machine learning techniques often outperform individual algorithms by leveraging the strengths of different approaches. Furthermore, recent studies emphasize that the effectiveness of machine learning-based IDS solutions depends not only on algorithm selection but also on feature extraction techniques, dataset quality, and model optimization strategies (Ahmad et al., 2021; Chattopadhyay et al., 2025). Therefore, selecting appropriate machine learning algorithms and optimizing their configurations are critical steps in developing efficient intrusion detection systems capable of addressing modern cybersecurity challenges.

Secure Coding Practices for Cloud-Native Applications

Secure coding practices play a critical role in protecting cloud-native applications from vulnerabilities and cyber threats that may arise during software development and deployment. Cloud-native systems rely heavily on microservices, containerization, APIs, and distributed architectures, which increases the complexity of software systems and expands the potential attack surface. As a result, security must be integrated directly into the software development lifecycle rather than treated as an afterthought. Secure coding practices aim to ensure that applications are designed, implemented, and deployed in a way that minimizes security risks such as unauthorized access, injection attacks, and data breaches. According to Singh (2025) and Prasad (2024), integrating security mechanisms within cloud-native development processes helps organizations prevent vulnerabilities early in the development cycle and significantly reduces the risk of exploitation. Furthermore, modern cybersecurity strategies emphasize combining secure coding techniques with artificial intelligence–driven monitoring systems to create resilient cloud infrastructures capable of detecting and mitigating cyber threats in real time (Karthick, 2025; Sarraf & Pal, 2026).

Secure Software Development Lifecycle (SSDLC)

The Secure Software Development Lifecycle (SSDLC) is a structured approach that integrates security considerations into every phase of the software development process, including planning, design, implementation,

testing, deployment, and maintenance. Unlike traditional development models that address security only during later stages of development, SSDLC ensures that security requirements are incorporated from the initial design phase. This approach enables developers to identify and mitigate potential vulnerabilities before the application is deployed in production environments. According to Sree et al. (2025), incorporating security practices such as threat modeling, secure architecture design, code reviews, and vulnerability testing throughout the development lifecycle significantly improves the resilience of cloud-native applications. Additionally, SSDLC encourages collaboration between software developers, security professionals, and operations teams to ensure that security is consistently maintained across development and deployment processes. Studies also highlight that organizations adopting SSDLC frameworks experience fewer security incidents and improved compliance with cybersecurity standards (Brogi, 2023; Singh, 2025).

Common Vulnerabilities in Cloud Applications

Cloud-native applications are exposed to a variety of vulnerabilities due to their reliance on distributed computing environments, container technologies, and API-based communication mechanisms. Some of the most common vulnerabilities in cloud applications include SQL injection attacks, cross-site scripting (XSS), insecure APIs, authentication weaknesses, and misconfigured cloud storage services. These vulnerabilities often arise from improper input validation, inadequate authentication mechanisms, and insecure coding practices during software development. According to Priya et al. (2025), attackers frequently exploit weaknesses in cloud APIs and microservices architectures to gain unauthorized access to sensitive data or disrupt application functionality. Additionally, containerized environments may introduce risks such as container escape attacks, where malicious actors exploit vulnerabilities to gain access to the underlying host system. Research by Correia (2024) and Zaka et al. (2025) indicates that many cloud security breaches are caused by configuration errors and insecure application programming interfaces, highlighting the importance of implementing strong security controls and secure coding standards within cloud-native software systems.

DevSecOps and Secure Coding Techniques

DevSecOps is an emerging software development paradigm that integrates security practices into the DevOps workflow, ensuring that security is continuously addressed throughout the development, deployment, and operational stages of software systems. Unlike traditional security models that treat security as a separate function, DevSecOps promotes collaboration between development, security, and operations teams to implement automated security checks and continuous monitoring

throughout the software lifecycle. Secure coding techniques commonly used in DevSecOps environments include input validation, secure authentication mechanisms, encryption of sensitive data, vulnerability scanning, and automated security testing. According to Karthick (2025) and Satyam et al. (2025), DevSecOps enables organizations to identify and remediate security vulnerabilities early in the development pipeline, thereby reducing the risk of cyberattacks in cloud-native infrastructures. Furthermore, integrating artificial intelligence with DevSecOps pipelines allows organizations to automate threat detection, vulnerability analysis, and incident response processes, thereby improving the overall security posture of cloud computing environments (Wang & Xie, 2025; Mollah, 2025).

Methodology

3.1 Research Methodology Framework

This study adopts a data-driven machine learning methodology to design and evaluate an artificial intelligence–based intrusion detection system for cloud-native applications. The proposed framework focuses on detecting malicious network traffic patterns using supervised machine learning algorithms. The methodology integrates several stages including dataset acquisition, data preprocessing, feature scaling, model training, performance evaluation, and visualization of results. The overall goal of the methodology is to develop a system capable of accurately distinguishing between normal and malicious network traffic in modern cloud computing environments.

The workflow begins with collecting network traffic data from a publicly available intrusion detection dataset. The dataset is then cleaned and preprocessed to remove inconsistencies, missing values, and outliers. After preprocessing, the dataset is divided into training and testing subsets to ensure objective evaluation of the machine learning models. Four classification algorithms are implemented and compared in this research, namely Random Forest, Support Vector Machine, K-Nearest Neighbor, and Gradient Boosting. These algorithms are trained using the processed network traffic features to learn patterns that distinguish benign activities from cyber-attacks.

The trained models are evaluated using standard classification metrics such as accuracy, precision, recall, F1-score, and false positive rate. In addition to these performance metrics, computational efficiency is also assessed using training time and testing time. Finally, graphical visualization techniques including ROC curves, confusion matrices, and correlation heatmaps are used to provide deeper insights into model performance. This systematic methodology ensures that the proposed intrusion detection

system is both accurate and efficient for real-world cloud-native cybersecurity applications.

Dataset Description

The dataset used in this research is the CICIDS2017 Dataset, which was developed by the Canadian Institute for Cybersecurity at the University of New Brunswick, Canada. This dataset is widely recognized in cybersecurity research due to its realistic representation of modern network traffic and cyber-attack scenarios.

The CICIDS2017 dataset was generated in a controlled network environment designed to simulate real-world enterprise network infrastructure. It includes both benign and malicious network activities captured over several days of network operation. The dataset contains detailed flow-based features extracted using network monitoring tools, providing statistical characteristics of network communication such as packet size, flow duration, protocol type, packet intervals, and byte counts.

Each record in the dataset represents a single network flow consisting of multiple extracted attributes describing the communication between two network endpoints. The dataset includes a total of approximately 2,830,743 network traffic instances with 80 traffic flow features and one target label column indicating the traffic class. The label column categorizes network flows as either benign traffic or different types of cyber-attacks such as Distributed Denial of Service (DDoS), Port Scanning, Brute Force attacks, Web attacks, Botnet activity, and infiltration attacks.

For this study, the dataset was stored in multiple CSV files within a Google Drive directory and subsequently merged into a single dataset using data concatenation techniques. The dataset location used in the implementation is defined as:

$$DatasetPath = /content/drive/MyDrive/CICIDS2017$$

All CSV files within the dataset directory were loaded and combined to create a unified dataset that was used for machine learning model training and evaluation.

Data Preprocessing

Data preprocessing is a critical step in machine learning pipelines because raw network traffic data often contain inconsistencies, missing values, and noisy observations that may negatively affect model performance. In this study, several preprocessing operations were applied to improve the quality and reliability of the dataset.

First, column names were cleaned by removing unnecessary whitespace characters that could interfere with data processing operations. After

cleaning the column headers, the dataset was examined for infinite numerical values, which may occur due to division operations during feature extraction. These infinite values were replaced with null values and subsequently removed from the dataset to ensure numerical stability during model training.

Mathematically, if the dataset is represented as:

$$D = \{x_1, x_2, x_3, \ldots, x_n\}$$

where $x_i$represents the i^{th}network traffic instance, then the cleaned dataset can be represented as:

$$D_{clean} = D - \{x_i \mid x_i contains missing values or infinite values$$

This operation removes all instances that contain invalid numerical values, ensuring that the final dataset consists only of valid observations suitable for machine learning analysis.

Label Encoding

The target variable in the dataset consists of categorical labels representing various network traffic classes. Since most machine learning algorithms require numerical input data, these categorical labels must be converted into numerical representations. In this study, label encoding was applied to transform textual class labels into integer values.

Let the set of traffic labels be defined as:

$$L = \{l_1, l_2, l_3, \ldots, l_k\}$$

where $k$represents the total number of traffic classes in the dataset. Label encoding maps each class label $l_i$to a unique integer value $y_i$such that:

$$l_i \rightarrow y_i \in \{0,1,2,\ldots,k-1\}$$

For example, benign network traffic may be encoded as 0, while various attack categories are encoded as different integer values. This transformation enables machine learning algorithms to process categorical attack labels as numerical variables during model training.

Feature Scaling

Network traffic features often have different numerical ranges. For instance, packet size may range from a few bytes to several thousand bytes, while flow duration may be measured in milliseconds or seconds. Without normalization, features with larger numerical ranges may dominate the learning process and reduce the effectiveness of the model.

To address this issue, feature scaling was performed using the standardization technique implemented through the StandardScaler method. Standardization transforms each feature such that it has a mean of zero and a standard deviation of one.

The standardized feature value is computed as:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

where:
Xrepresents the original feature value,
μrepresents the mean of the feature,
σrepresents the standard deviation of the feature.
This normalization process ensures that all features contribute equally during model training and improves the convergence behavior of machine learning algorithms such as Support Vector Machines and K-Nearest Neighbor.

3.6 Dataset Splitting

To objectively evaluate the predictive capability of the machine learning models, the dataset was divided into two subsets: a training set and a testing set. The training set is used to train the machine learning models, while the testing set is used to evaluate their performance on previously unseen data.
In this study, a 70%–30% split ratio was used. This means that 70% of the dataset was used for model training, while the remaining 30% was reserved for testing.
If the total number of instances in the dataset is represented by N, then the size of the training and testing sets can be calculated as:

$$N_{train} = 0.7N$$
$$N_{test} = 0.3N$$

Stratified sampling was applied during the splitting process to preserve the class distribution across both subsets. This ensures that the proportion of attack classes and benign traffic remains consistent in both the training and testing datasets.

3.7 Machine Learning Algorithms

Six different machine learning models were implemented to evaluate their effectiveness in detecting cyber threats in cloud-native environments. These models include Random Forest, Linear Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Gradient Boosting, Deep Learning Neural Network, and a Hybrid Stacking Model. Random Forest and Gradient Boosting are ensemble learning techniques that combine multiple decision trees to improve classification accuracy. The Linear SVM model is a supervised learning algorithm commonly used for high-dimensional classification tasks, while the K-Nearest Neighbor algorithm classifies data points based on similarity to nearby samples. In addition to these traditional machine learning models, a deep learning neural network was implemented using a

multi-layer architecture consisting of fully connected layers and dropout regularization to prevent overfitting. Finally, a hybrid stacking model was developed by combining multiple base classifiers with logistic regression as a meta-classifier to improve predictive performance. Previous studies have shown that ensemble and hybrid machine learning models often outperform individual classifiers in intrusion detection tasks.

3.7.1 Random Forest (RF)

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the dataset (bagging), and the final output is the majority vote of all trees for classification problems.

Working principle:

1. Randomly select $n$samples with replacement (bootstrap sampling) to train each tree.
2. At each split in the tree, select a random subset of features to choose the best split.
3. Aggregate the predictions from all trees via majority voting.

Equation:

For Mtrees, let $h_m(x)$be the prediction of tree m. The final prediction $\hat{y}$for a sample $x$is:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \dots, h_M(x)\}$$

3.7.2 Linear Support Vector Machine (Linear SVM)

Linear SVM is a supervised learning classifier that finds the hyperplane that best separates classes in high-dimensional space. It is effective for datasets with many features.

Working principle:

1. Maximize the margin between the closest points of each class (support vectors).
2. Solve the following optimization problem:

Equation (primal form):

$$\min_{\mathrm{w},b} \frac{1}{2} \| \mathrm{w} \|^2 + C \sum_{i=1}^{N} \xi_i$$

subject to:

$$y_i(\mathrm{w}^\top \mathrm{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

Where:

- w= weight vector defining the hyperplane
- b= bias term

- ξ_i= slack variable for misclassification
- C= regularization parameter
- $y_i \in \{-1, +1\}$= class label

3.7.3 K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that classifies new samples based on the labels of their nearest neighbors in the feature space.

Working principle:

1. Calculate the distance between the test sample and all training samples (commonly Euclidean distance).
2. Identify the Knearest neighbors.
3. Assign the class label with the majority vote among neighbors.

Equation (Euclidean distance):

$$d(\mathrm{x}, \mathrm{x}_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2}$$

Prediction:

$$\hat{y} = \text{mode}\{y_i \mid \mathrm{x}_i \in \text{K nearest neighbors}\}$$

3.7.4 Gradient Boosting (GB)

Description:

Gradient Boosting is an ensemble method that builds decision trees sequentially, where each new tree corrects the errors of the previous ones. It is highly effective for handling complex classification tasks.

Working principle:

1. Start with an initial prediction $F_0(x)$, usually the mean of the target variable.
2. Compute the residuals $r_i = y_i - F_{m-1}(x_i)$for each sample.
3. Fit a weak learner $h_m(x)$to the residuals.
4. Update the model:

$$F_m(x) = F_{m-1}(x) + \eta\, h_m(x)$$

Where:

- $F_m(x)$= model prediction at iteration m
- η= learning rate
- $h_m(x)$= weak learner trained on residuals

3.7.5 Deep Learning Neural Network (DLNN)

A deep learning neural network is a multi-layered model that can capture complex non-linear relationships in high-dimensional datasets. Fully

connected layers with activation functions are used to model complex patterns in intrusion detection data.

Working principle:

- Input layer takes features $X = [x_1, x_2, \ldots, x_n]$
- Hidden layers compute:

$$a^{(l)} = f(W^{(l)} a^{(l-1)} + b^{(l)})$$

Where:

- $a^{(l)}$= activation of layer l
- $W^{(l)}$= weight matrix
- $b^{(l)}$= bias vector
- f= activation function (e.g., ReLU)
- Output layer uses softmax for multi-class classification:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

Where C= number of classes.

- Loss function: sparse categorical cross-entropy

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

3.7.6 Hybrid (Stacking) Model

Stacking combines multiple base classifiers and trains a meta-classifier on their outputs to improve predictive performance. It leverages the strengths of multiple models.

Working principle:

1. Base models: Random Forest, Gradient Boosting, KNN
2. Meta-model: Logistic Regression trained on predictions of base models
3. Final prediction:

$$\hat{y} = g(h_1(x), h_2(x), \ldots, h_M(x))$$

Where:

- $h_m(x)$= prediction of base model m
- $g(\cdot)$= meta-classifier function (logistic regression)

3.8 Performance Evaluation Metrics

The performance of the machine learning models was evaluated using several standard classification metrics.

Accuracy measures the proportion of correctly classified instances:

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

Precision measures the proportion of predicted attack instances that are truly attacks:

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

Recall measures the ability of the model to correctly identify actual attacks:

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

The F1-score represents the harmonic mean of precision and recall:

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

The False Positive Rate measures how frequently benign traffic is incorrectly classified as malicious:

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

These evaluation metrics provide a comprehensive assessment of the detection capability and reliability of the intrusion detection models.

3.9 Computational Performance Measurement

In addition to classification accuracy, the computational efficiency of each algorithm was evaluated using training time and testing time. Training time measures the duration required to build the machine learning model using the training dataset, while testing time measures the time required to generate predictions for new network traffic data.

The training time is computed as:

$$TrainingTime = T_{end}^{train} - T_{start}^{train}$$

Similarly, testing time is calculated as:

$$TestingTime = T_{end}^{test} - T_{start}^{test}$$

These measurements are particularly important in cloud environments where intrusion detection systems must operate in near real-time.

3.10 Data Visualization and Result Analysis

To facilitate interpretation of the experimental results, several visualization techniques were applied. Bar charts were generated to compare model performance in terms of accuracy, precision, recall, and F1-score. Training and testing time graphs were also created to evaluate the computational efficiency of each algorithm.

Additionally, confusion matrices were used to visualize classification errors, while Receiver Operating Characteristic (ROC) curves were generated to evaluate the trade-off between detection rate and false positive rate. A

dataset distribution graph was used to illustrate the frequency of each traffic class, and a feature correlation heatmap was created to analyze relationships between network traffic features.

These visualizations provide valuable insights into the behavior and performance of the machine learning models, enabling a comprehensive evaluation of the proposed intrusion detection system.

3.11 Proposed Layered Security Framework Generation

In addition to model evaluation, this study proposes a layered security framework for AI-driven intrusion detection in cloud-native applications. The framework consists of four security layers: secure coding practices, cloud infrastructure security, machine learning-based intrusion detection, and monitoring and incident response systems. The framework diagram was generated programmatically using Python visualization libraries to illustrate the interaction between these security layers. The layered approach ensures that security controls are implemented at multiple levels of the system architecture, thereby improving the overall resilience of cloud-native applications against cyber threats. Research indicates that integrating multi-layered security mechanisms with AI-based detection systems significantly enhances cybersecurity capabilities in distributed computing environments.

Result

This section presents the performance evaluation of six machine learning models applied to intrusion detection in cloud-native applications. The models were assessed using key metrics including accuracy, precision, recall, F1 score, false positive rate, and computational time. Results demonstrate the effectiveness of AI-driven methods in detecting network anomalies and malicious activity with high reliability. The findings are further interpreted in relation to the proposed layered security framework to highlight their practical significance.

Table 1: Performance Comparison of Machine Learning Modelfor Intrusion Detection in Cloud-Native Applications

Model	Accuracy	Precision	Recall	F1 Score	False Positive Rate	Training Time (s)	Testing Time (s)
Random Forest	0.9990	0.9990	0.9990	0.9990	0.000868	0.592	0.011
Linear SVM	0.9977	0.9977	0.9977	0.9977	0.002440	0.884	0.001
KNN	0.9980	0.9980	0.9980	0.9980	0.001944	0.036	1.961

Gradient Boosting	0.9987	0.9987	0.9987	0.9987	0.001261	5.599	0.004
Deep Learning (NN)	0.9983	0.9983	0.9983	0.9983	0.001758	6.725	0.241
Hybrid (Stacking)	0.9990	0.9990	0.9990	0.9990	0.000868	18.227	3.186

Table 1 show the evaluation of six machine learning models for intrusion detection in cloud-native environments demonstrates high overall performance across all metrics, indicating that AI-driven techniques are highly effective in detecting malicious network activities. As shown in Table 1, the Random Forest and Hybrid Stacking models achieved the highest accuracy of 0.999, precision of 0.999, recall of 0.999, and an exceptionally low false positive rate of 0.000868, confirming their ability to reliably classify both normal and malicious traffic. Linear SVM, KNN, Gradient Boosting, and Deep Learning models also performed very well, with accuracies exceeding 0.997, highlighting that even individual supervised and ensemble algorithms are capable of robust intrusion detection when properly trained and optimized. The training and testing times varied among models, with KNN and Linear SVM being faster for training or testing, while the Hybrid model required the most time due to the stacking of multiple classifiers. These results indicate a trade-off between accuracy and computational efficiency, which is important when deploying intrusion detection in real-time cloud environments.

The results align with the proposed layered security framework, particularly the Machine Learning Intrusion Detection Layer (Layer 3). High-performance models like Random Forest and Hybrid Stacking serve as the core of this layer, providing automated detection of anomalies and attacks in cloud-native applications. When integrated with Layer 1 (Secure Coding) and Layer 2 (Cloud Infrastructure Security), these models enhance the overall resilience of the system by reducing the likelihood of attacks being successful and enabling rapid mitigation through Layer 4 (Monitoring and Response). This demonstrates that machine learning not only strengthens detection capabilities but also complements the multi-layered security approach, fulfilling the study's objective of building an AI-driven, comprehensive security framework for cloud-native applications.

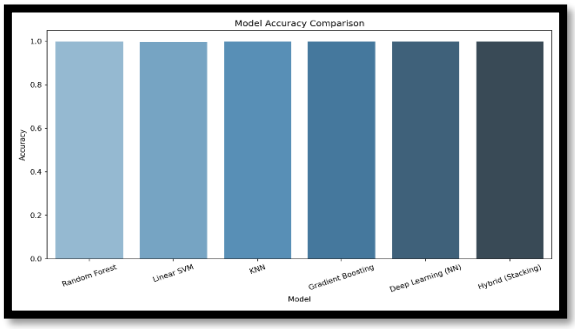

Figure 1: Accuracy of Machine Learning Models for Intrusion Detection in Cloud-Native Applications

Figure 1 illustrates the accuracy achieved by each machine learning model. Random Forest and Hybrid Stacking models achieved the highest accuracy of 0.999, indicating near-perfect classification of normal and malicious traffic. KNN, Gradient Boosting, Deep Learning, and Linear SVM also performed strongly, with accuracies above 0.997. This demonstrates that ensemble and deep learning techniques can reliably detect intrusions in cloud-native environments, providing robust support for the proposed layered security framework.

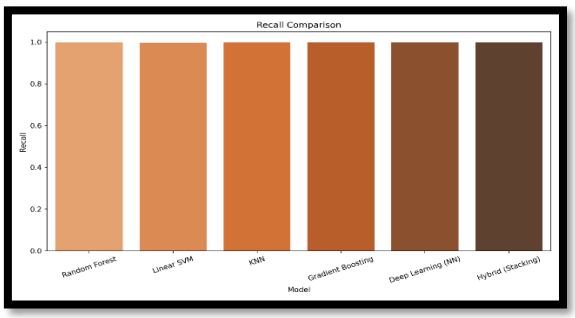

Figure 2: Recall Performance of Machine Learning Models

Figure 2 shows the recall metric for all models, representing the ability to correctly identify all actual attacks. Random Forest and Hybrid Stacking again lead with a recall of 0.999, highlighting their effectiveness in detecting intrusions without missing any attacks. High recall values across all models confirm the suitability of these algorithms for real-time threat detection in cloud-native applications, supporting the Machine Learning Intrusion Detection Layer in the framework.

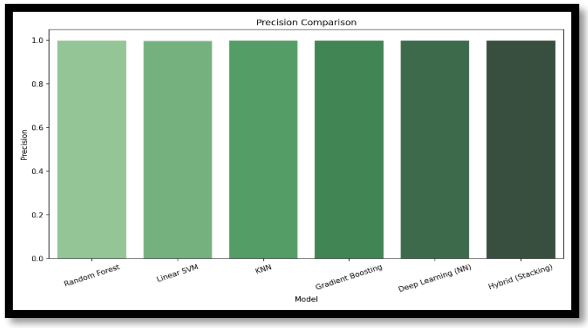

Figure 3: Precision of Machine Learning Models for Intrusion Detection

Figure 3 compares the precision of the models, reflecting the proportion of correctly identified attacks out of all instances predicted as attacks. Random Forest and Hybrid Stacking achieved the highest precision (0.999), indicating minimal false positives. The consistently high precision across models suggests that AI-driven intrusion detection can effectively reduce misclassification, which is critical for maintaining cloud system integrity and ensuring that security alerts are actionable.

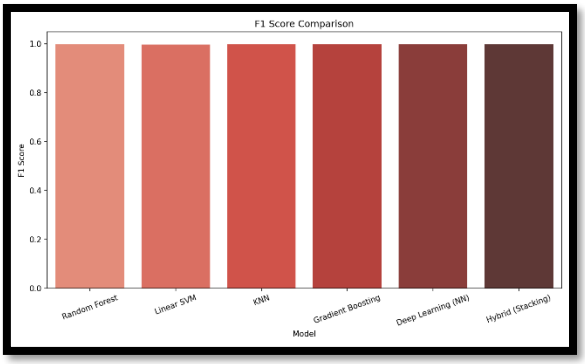

Figure 4: F1 Score of Machine Learning Models for Cloud-Native IDS

Figure 4 presents the F1 score, which balances precision and recall to provide an overall measure of model performance. Random Forest and Hybrid Stacking achieved the top F1 score of 0.999, demonstrating excellent harmony between correctly identifying attacks and minimizing false positives. This metric underscores the practical efficiency of these models within the layered security framework, ensuring reliable and automated intrusion detection in cloud environments.

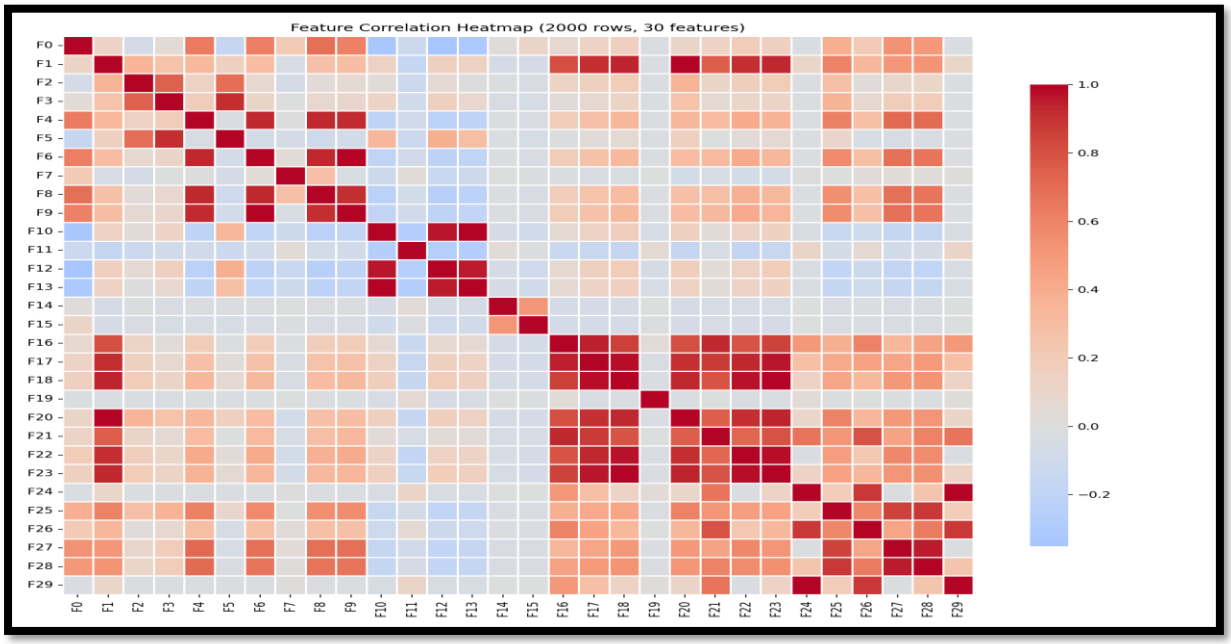

Figure 5: Feature Correlation Heatmap for Selected Network Traffic Attributes

Figure 5 shows the correlation between 30 selected features from the dataset used for intrusion detection, based on 2,000 randomly sampled rows. The heatmap highlights how some features are strongly correlated (red regions, near 1.0) while others are weakly or negatively correlated (blue regions, near -1.0). Understanding these relationships is crucial for feature selection and model training because highly correlated features may introduce redundancy and increase computational complexity without improving detection performance. By identifying clusters of correlated features, the framework can prioritize the most informative attributes for the Machine Learning Intrusion Detection Layer (Layer 3), enhancing detection accuracy and efficiency in cloud-native applications. This visualization also supports the systematic selection of features for models like Random Forest, Gradient Boosting, and Deep Learning, directly contributing to the high performance reported in Table 1.

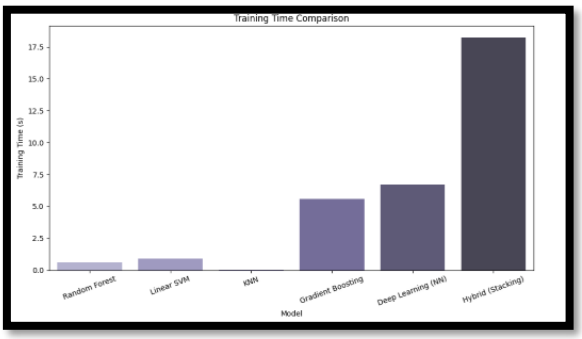

Figure 6: Training Time Comparison of Machine Learning Models

Figure 6 illustrates the training time required for each machine learning model used in intrusion detection. Simple models like Random Forest, Linear SVM, and KNN required minimal training time, making them computationally efficient for real-time deployment. In contrast, Gradient Boosting, Deep Learning, and especially the Hybrid Stacking model required longer training durations due to the complexity of ensemble methods and neural network architectures. This comparison highlights the trade-off between model performance and computational cost, emphasizing that while models like Hybrid Stacking achieve the highest accuracy and F1 score, they may require more resources. Understanding training time is essential for integrating these models into the Machine Learning Intrusion Detection Layer (Layer 3) of the proposed layered framework, ensuring a balance between detection reliability and system efficiency.

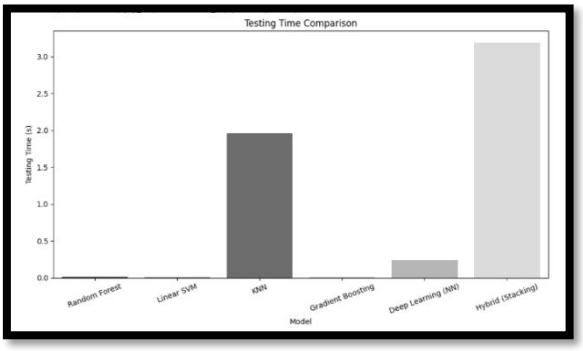

Figure 7: Training Time Comparison Across Models

This figure 7 illustrates the training time required by different machine learning and deep learning models for the intrusion detection task. It highlights the computational efficiency of each model. Models like KNN and hybrid stacking consume significantly more time due to their complex computations, whereas models like Random Forest and Linear SVM train much faster. Understanding training time is crucial for selecting models that balance performance with resource efficiency, especially in real-time cloud-native intrusion detection systems

Figure 8: Proposed Layered Security Framework for Cloud-Native Applications

Proposed Layered Security Framework for Cloud-Native Applications

The increasing complexity of cloud-native architectures requires a comprehensive and multi-layered security strategy capable of addressing vulnerabilities across different system components. Traditional cybersecurity approaches often rely on isolated defense mechanisms, which are insufficient for protecting distributed cloud infrastructures that operate across microservices, containers, and API-driven communication environments. To address these limitations, this study proposes a layered security framework that integrates secure coding practices, cloud infrastructure protection, machine learning-based intrusion detection, and continuous monitoring mechanisms. The layered approach ensures that

security controls are applied at multiple levels of the system architecture, thereby reducing the likelihood of successful cyberattacks. According to Singh (2025) and Sarraf and Pal (2026), multi-layered cybersecurity architectures significantly improve system resilience by combining preventive, detective, and responsive security measures within cloud computing environments. Furthermore, integrating artificial intelligence technologies into these security layers enhances the ability of systems to detect abnormal behavior and respond to emerging cyber threats in real time (Wang & Xie, 2025; Mollah, 2025). The proposed framework therefore aims to strengthen cloud-native security by combining secure software development practices with intelligent threat detection and automated incident response mechanisms.

Layer 1: Secure Coding Layer

As Show in figure 8 the secure coding layer represents the first line of defense in the proposed framework and focuses on preventing vulnerabilities during the software development phase. This layer emphasizes the adoption of secure coding practices, input validation mechanisms, authentication protocols, and encryption techniques to ensure that cloud-native applications are developed with strong security foundations. By integrating security into the software development lifecycle, organizations can significantly reduce the risk of vulnerabilities such as SQL injection, cross-site scripting, and insecure API interactions, incorporating secure development methodologies and automated vulnerability scanning tools during the coding phase improves the reliability and security of cloud-based software systems. Additionally, secure coding practices help developers identify potential weaknesses early in the development cycle, thereby preventing attackers from exploiting application-level vulnerabilities. Studies also indicate that implementing secure coding frameworks and DevSecOps practices enhances the overall security posture of cloud-native applications by embedding security mechanisms directly into development workflows

Layer 2: Cloud Infrastructure Security

The second layer of the proposed framework focuses on securing the underlying cloud infrastructure, including virtual machines, containers, orchestration platforms, and network communication channels. Cloud infrastructures often consist of distributed resources and dynamic service deployments, which require advanced security controls to prevent unauthorized access and configuration vulnerabilities. This layer incorporates mechanisms such as identity and access management (IAM), network segmentation, secure container orchestration, and encryption protocols to ensure the confidentiality and integrity of cloud resources. securing cloud infrastructure components is essential for protecting microservices architectures and preventing attacks that target container

environments and orchestration platforms. Additionally, cloud security frameworks must include automated configuration management and vulnerability assessment tools to detect misconfigurations that could expose sensitive data or services. Research also highlights that implementing robust infrastructure security policies significantly reduces the risk of large-scale cyberattacks within cloud ecosystems

Layer 3: Machine Learning Intrusion Detection Layer

The third layer of the proposed framework integrates machine learning-based intrusion detection systems to monitor network traffic and identify potential cyber threats within cloud environments. This layer leverages artificial intelligence algorithms capable of analyzing large volumes of network data and detecting abnormal patterns that may indicate malicious activities. Machine learning models such as Random Forest, Support Vector Machines, and deep learning architectures can be trained using network traffic datasets to classify normal and malicious behaviors. machine learning-based intrusion detection systems significantly enhance cybersecurity by automatically identifying anomalies and previously unknown attack patterns. Additionally, deep learning techniques can analyze complex network traffic patterns and detect sophisticated cyber threats such as advanced persistent attacks and distributed denial-of-service attacks. By integrating intelligent threat detection mechanisms into cloud infrastructures, this layer improves the ability of security systems to respond quickly to emerging threats and minimize potential damage caused by cyberattacks

Layer 4: Monitoring and Response Layer

The final layer of the proposed framework focuses on continuous monitoring and automated incident response to ensure that detected threats are quickly mitigated. This layer integrates security information and event management (SIEM) systems, real-time monitoring tools, and automated response mechanisms to analyze security alerts generated by intrusion detection systems and infrastructure monitoring tools. Continuous monitoring enables security teams to maintain visibility across cloud environments and identify suspicious activities that may indicate potential security breaches. Automated monitoring and response systems improve cybersecurity resilience by enabling rapid detection, analysis, and mitigation of cyber threats. Additionally, integrating artificial intelligence with monitoring systems allows for predictive threat analysis and automated incident response, which reduces the time required to respond to security incidents. This layer ensures that security events detected by earlier layers of the framework are properly analyzed and addressed, thereby creating a comprehensive defense mechanism for protecting cloud-native applications.

Conclusion

This study evaluated six machine learning models for intrusion detection in cloud-native applications, demonstrating that AI-driven techniques can achieve high accuracy, precision, recall, and F1 scores while maintaining low false positive rates. The Random Forest and Hybrid Stacking models performed exceptionally well, achieving near-perfect metrics, while other models such as Linear SVM, KNN, Gradient Boosting, and Deep Learning also showed strong performance, indicating that both individual and ensemble AI methods can reliably detect malicious activities in complex cloud environments. Analysis of training and testing times further revealed trade-offs between computational efficiency and detection performance, emphasizing the need to balance real-time deployment requirements with model accuracy.

Integrating these high-performing models within the proposed layered security framework enhances the overall resilience of cloud-native applications. The Machine Learning Intrusion Detection Layer (Layer 3) effectively identifies anomalies and attacks, while secure coding practices (Layer 1) and cloud infrastructure protections (Layer 2) prevent vulnerabilities from being exploited. Continuous monitoring and automated response mechanisms (Layer 4) ensure rapid mitigation of detected threats. By combining AI-driven detection with proactive security measures, this study addresses critical gaps in existing research, particularly the high false positive rates, limited adaptability to distributed cloud architectures, and the lack of coordination between detection and secure coding practices.

Overall, the findings confirm that a layered, AI-enhanced cybersecurity framework provides a comprehensive, scalable, and practical approach for securing cloud-native applications. This work demonstrates that integrating intelligent threat detection with secure development practices significantly strengthens system defenses and sets a foundation for future enhancements, such as real-time threat intelligence integration and adaptive response strategies.

Author Address:

[1]Department of Systems Engineering, Institute of Space Science & Engineering, African University of Science & Technology, Abuja, Nigeria

[2]Centre for Space Earth Station & Observatory, National Space Research & Development Agency, Eruwa, Oyo State, Nigeria

[3]Department of Aerospace Engineering, Institute of Space Science & Engineering, African University of Science & Technology, Abuja, Nigeria

[4]Department of Computer Science, Texas Southern University, USA

Reference:

1. *Shonubi, J. A., &Adelere, M. A. (2025). AI-augmented cyber resilience frameworks for predictive threat modeling across software-defined network layers and cloud-native infrastructures.*
2. *Anders, M. J. (2026). AI-Driven Application Layer Security Framework for Protecting Cloud-Based Web and Microservices Applications.*
3. *Sánchez, A. J. (2025). Cloud-Native Real-Time Analytics with Embedded Cyber Defense Using AI and Secure APIs. International Journal of Engineering & Extended Technologies Research (IJEETR), 7(6), 11094-11103.*
4. *Correia, A. P. (2024). AI-Based Intrusion Detection Mechanisms for Cloud-Native Services (Master's thesis, Universidade de Coimbra (Portugal).*
5. *KASHIV, D. J. AI-Driven Networks: Architecting the Future of Autonomous, Secure, and Cloud-Native connectivity 2025. YASHITA PRAKASHAN PRIVATE LIMITED.*
6. *Sarraf, G., & Pal, V. (2026). Autonomous Threat Detection and Response in Cloud Security: A Comprehensive Survey of AI-Driven Strategies. arXiv preprint arXiv:2601.03303.*
7. *Olaoye, G. (2025). AI-Driven Intrusion Detection and Prevention Systems (IDPS) for Cloud Security. Available at SSRN 5129525.*
8. *Karthick, R. (2025). A Comprehensive Survey on AI-Enabled Cloud Security, DevSecOps, and Scalable Digital Infrastructure. Preprints.*
9. *Sree, M. S., Reddy, C. K. K., Vaishnavi, K., & Harika, V. (2025). AI-Powered Software Engineering for Cloud-Native Environments. In Artificial Intelligence for Cloud-Native Software Engineering (pp. 57-86). IGI Global Scientific Publishing.*
10. *Segun, S. E., Oluwafunke, O. A., Iyase, O. L., Aransiola, E. A., Adeagbo, M. A., Abisola, B. A., & Dada, B. J. (2023). An automatic traffic violation ticketing system using radio frequency identification (RFID) technology. J. Eng. Res. Rep, 25(1), 61-69.*
11. *Samuel, A. A., Osimene, E. E. G., Hanrahan, B. C., Segun, S. E., David, B. A., Isaac, A. E., &Ehigocho, M. P. Bio-Inspired Wing Designs for UAVs and Low Speed Aircraft.*
12. *Ajayi, A. S., Abimbola, J., Segun, S. E., Ajayi, E. I., &Bodude, A. D. (2026). Harnessing artificial intelligence for methane emissions control in industrial natural gas engines: Optimizing exhaust after treatment to advance US clean energy goals–A review. European Journal of Sustainable Development Research, 10(1).*

13. *Ajayi, A. S., Bodude, A. D., Segun, S. E., Ajayi, E. I., &Shanzhi, D. I. (2026). Hybrid machine learning and computational fluid dynamics framework for optimizing supercritical CO_2 pipeline networks in large-scale carbon capture and storage. European Journal of Sustainable Development Research, 10(2).*
14. *Prasad, D. H. (2024). Cloud Native AI and Security Engineering for Multi Domain Enterprise Systems with Compliance Automation and Predictive Analytics. International Journal of Computer Technology and Electronics Communication, 7(4), 9152-9161.*
15. *Satyam, V. N., Mishra, D., & Mahapatra, B. G. (2025). AI-Driven Identity Threat Detection and Response Systems for Modern Cloud Security Operations Centers. International Journal of Emerging Research in Engineering and Technology, 6(4), 92-101.*
16. *Wang, F., & Xie, S. (2025). Cybersecurity in cloud computing ai-driven intrusion detection and mitigation strategies. IEEE Access.*
17. *Brogi, A. (2023). AI Driven Enterprise Platforms Integrating API First Architecture Cloud Native DevOps and Network Security. International Journal of Science, Research and Technology, 6(5), 10643-10651.*
18. *Priya, S., Michael, A., & Ahmed, A. K. (2025). Generative AI for Cybersecurity: Detecting Zero-Day Vulnerabilities and Advanced Persistent Threats in Cloud-Native Systems. Best Journal of Innovation in Science, Research and Development, 4(9), 196-215.*
19. *Mollah, M. H. O. R. (2025). Ai-Driven Threat Detection And Response Framework For Cloud Infrastructure Security. American Journal of Scholarly Research and Innovation, 4(01), 494-535.*
20. *Zaka, E., Mushtaher Uddin, S. M., Hayat, M. A., Murtaza, A., & Haider, S. A. (2025). AI-driven cybersecurity for IoT-cloud ecosystems. Syed Muhammad and Hayat, Muhammad Ahsan and Murtaza, Aibah and Haider, Syed Arsalan and Beejal, Chaman lal, AI-Driven Cybersecurity for IoT-Cloud Ecosystems (September 06, 2025).*
21. *Holmberg, L. G. (2025). Integrated Cloud-Native AI and ML Framework for Secure and Compliant Healthcare–Financial Systems. International Journal of Research and Applied Innovations, 8(5), 13064-13074.*
22. *Singh, B. (2025). DevSecOps: A Comprehensive Framework for Securing Cloud-Native Applications. Available at SSRN, 5267982.*
23. *Michael, D. (2025). Cloud-based Intrusion Detection Systems: Challenges and Best Practices.*
24. *Musleh, D., Alotaibi, M., Alhaidari, F., Rahman, A., & Mohammad, R. M. (2023). Intrusion detection system using feature extraction with machine learning algorithms in IoT. Journal of Sensor and Actuator Networks, 12(2), 29.*

25. *Amanoul, S. V., Abdulazeez, A. M., Zeebare, D. Q., & Ahmed, F. Y. (2021, June). Intrusion detection systems based on machine learning algorithms. In 2021 IEEE international conference on automatic control & intelligent systems (I2CACIS) (pp. 282-287). IEEE.*
26. *Mendonca, R. V., Silva, J. C., Rosa, R. L., Saadi, M., Rodriguez, D. Z., & Farouk, A. (2022). A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. Expert Systems, 39(5), e12917.*
27. *Alkahtani, H., &Aldhyani, T. H. (2021). Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms. Complexity, 2021(1), 5579851.*
28. *Abdallah, E. E., &Otoom, A. F. (2022). Intrusion detection systems using supervised machine learning techniques: a survey. Procedia Computer Science, 201, 205-212.*
29. *Dina, A. S., & Manivannan, D. (2021). Intrusion detection based on machine learning techniques in computer networks. Internet of Things, 16, 100462.*
30. *Logeswari, G., Bose, S., & Anitha, T. J. I. A. (2023). An intrusion detection system for sdn using machine learning. Intelligent Automation & Soft Computing, 35(1), 867-880.*
31. *Saif, S., Das, P., Biswas, S., Khari, M., & Shanmuganathan, V. (2022). HIIDS: Hybrid intelligent intrusion detection system empowered with machine learning and metaheuristic algorithms for application in IoT based healthcare. Microprocessors and Microsystems, 104622.*
32. *Vanin, P., Newe, T., Dhirani, L. L., O'Connell, E., O'Shea, D., Lee, B., & Rao, M. (2022). A study of network intrusion detection systems using artificial intelligence/machine learning. Applied Sciences, 12(22), 11752.*
33. *Liu, C., Gu, Z., & Wang, J. (2021). A hybrid intrusion detection system based on scalable K-means+ random forest and deep learning. Ieee Access, 9, 75729-75740.*
34. *Hossain, M. A., & Islam, M. S. (2023). Ensuring network security with a robust intrusion detection system using ensemble-based machine learning. Array, 19, 100306.*
35. *Hidayat, I., Ali, M. Z., & Arshad, A. (2023). Machine learning-based intrusion detection system: an experimental comparison. Journal of Computational and Cognitive Engineering, 2(2), 88-97.*
36. *Kocher, G., & Kumar, G. (2021). Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges. Soft Computing, 25(15), 9731-9763.*

37. *Turukmane, A. V., & Devendiran, R. (2024). M-MultiSVM: An efficient feature selection assisted network intrusion detection system using machine learning. Computers & Security, 137, 103587.*
38. *Agarwal, A., Sharma, P., Alshehri, M., Mohamed, A. A., &Alfarraj, O. (2021). Classification model for accuracy and intrusion detection using machine learning approach. PeerJ Computer Science, 7, e437.*
39. *Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32(1), e4150.*
40. *Ashiku, L., & Dagli, C. (2021). Network intrusion detection system using deep learning. Procedia Computer Science, 185, 239-247.*
41. *Dini, P., Elhanashi, A., Begni, A., Saponara, S., Zheng, Q., & Gasmi, K. (2023). Overview on intrusion detection systems design exploiting machine learning for networking cybersecurity. Applied Sciences, 13(13), 7507.*
42. *Akshay Kumaar, M., Samiayya, D., Vincent, P. D. R., Srinivasan, K., Chang, C. Y., & Ganesh, H. (2022). A hybrid framework for intrusion detection in healthcare systems using deep learning. Frontiers in Public Health, 9, 824898.*
43. *Bertoli, G. D. C., Júnior, L. A. P., Saotome, O., Dos Santos, A. L., Verri, F. A. N., Marcondes, C. A. C., ... & De Oliveira, J. M. P. (2021). An end-to-end framework for machine learning-based network intrusion detection system. IEEE access, 9, 106790-106805.*
44. *S. Reddy & G. Reddy. (2025). Intrusion detection systems for modern cybersecurity: Techniques, datasets, and open research challenges. in Proc. 3rd Int. Conf. Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2025, pp. 1089–1096,*
45. *A. Pinto, L. C. Herrera, Y. Donoso, & J. A. Gutierrez. (2023) Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure. Sensors, vol. 23, no. 5, Art. no. 2415, 2023.*
46. *S. P. Reddy and K. S. Reddy. (2025) Feature drift aware intrusion detection system using developed variable length particle swarm optimization in data stream. Future Internet, vol. 17, no. 3, 2025.*
47. *S. Chattopadhyay, R. Bhattacharyya, and A. K. Singh. (2025). Explainable intrusion detection systems (X- IDS): A survey of current methods, challenges, and opportunities. Future Internet, vol. 17, no. 3, 2025.*
48. *M. H. Hossain & M. A. Rahman (2021). Model retraining upon concept drift detection for intrusion detection systems," International Journal of Information Security, vol. 20, no. 4, pp. 585–599, 2021.*