AI-Based Network Threat Detection using Wavelet Transform

Vinit Venkanna Guglot[1], Yash Anand Ghotekar[2], Gourav Pramod Kumbhare[3], Dikshant Vilas Fulzele[4], Sarang Vinod Channe[5], Prof.Sachin dhawas[6]
[1,2,3,4,5]*Department of CSE Student of RCERT Chandrapur, Maharashtra India,*
[6]*Department of CSE, Professor of RCERT Chandrapur, Maharashtra, India*

***Abstract*— Artificial Intelligence and Machine Learning have become crucial technologies for securing modern networks and managing data traffic over the internet. However, the increasing sophistication of cyber-attacks creates significant security challenges, such as unauthorized access and network intrusions. This project, titled "AI-Based Network Threat Detection using Wavelet Transform," focuses on developing an advanced machine learning-based intrusion detection system to identify malicious activities by leveraging signal processing techniques.**

The proposed system uses the Discrete Wavelet Transform (DWT) to decompose complex network traffic data into multi-resolution time-frequency coefficients, allowing the system to isolate anomalies and sudden traffic fluctuations from normal behaviour. Publicly available benchmark datasets, such as NSL-KDD and CICIDS2017, are utilized for training and evaluating the system's performance. The project aims to improve threat detection accuracy and reduce false alarms by combining wavelet-based feature extraction with intelligent machine learning classifiers.

The system is implemented in a local development environment using Python and Visual Studio Code (VS Code) for academic purposes. It does not perform live intrusion detection on real-time networks but demonstrates the practical application of signal processing and machine learning in network security. The project helps in understanding advanced feature engineering pipelines and the implementation of machine learning techniques in cybersecurity applications.

***Index Terms*—Artificial Intelligence, Network Threat Detection, Discrete Wavelet Transform (DWT), Feature Extraction, Machine Learning, Visual Studio Code (VS Code), Cybersecurity Simulation.**

I. INTRODUCTION

1.1 Introduction

In the modern digital era, computer networks and internet-based services have become an essential part of daily life. Organizations, industries, educational institutions, banks, healthcare systems, and government sectors heavily depend on network communication for storing, processing, and transferring data. As the use of digital technologies continues to increase, cyber threats and network attacks are also growing rapidly. Attackers continuously develop new methods to exploit network vulnerabilities, steal sensitive information, disrupt services, and damage organizational systems. Therefore, maintaining network security has become one of the most important challenges in the field of information technology.

Among various cyberattacks, Distributed Denial of Service (DDoS) attacks and Port Scanning attacks are considered highly dangerous and common threats. A DDoS attack floods a target server or network with a huge amount of traffic, causing the system to become slow or unavailable to legitimate users. Similarly, attackers use Port Scanning to identify open ports and vulnerabilities in a network before launching further attacks. These attacks can lead to monetary loss, service interruption, data leakage, and reduced system performance. Traditional security systems such as firewalls and signature-based intrusion detection systems are often unable to detect advanced and evolving attack patterns effectively.

To overcome these limitations, Artificial Intelligence (AI) and Machine Learning (ML) technologies are increasingly being used in cybersecurity applications. Machine learning algorithms can analyze substantial

amounts of network traffic data, learn traffic patterns, and identify suspicious or malicious activities automatically. AI-based systems provide faster detection, improved accuracy, reduced human effort, and the ability to handle complex cyber threats efficiently. These intelligent systems can continuously improve their performance by learning from historical data and detecting unknown attack behaviours.

This project focuses on developing an AI-Based Threat Detection System for identifying cyberattacks such as DDoS and PortScan attacks using machine learning techniques. The system uses the CICIDS2017 dataset, which contains realistic network traffic data including normal and attack traffic. Data preprocessing techniques such as cleaning, normalization, feature extraction, and train-test splitting are applied to improve the quality of the dataset. Machine learning algorithms are then trained on the processed data to classify network traffic into distinct categories such as Benign, DDoS, and PortScan traffic.

The proposed system aims to provide accurate and efficient detection of cyber threats by analyzing network traffic patterns. The project also demonstrates the importance of AI in modern cybersecurity solutions and highlights how intelligent systems can improve network protection mechanisms. This research can be further extended for real-time threat detection, cloud security, and large-scale enterprise network monitoring systems.

1.2 Background

With the rapid growth of internet usage and digital communication, cyberattacks such as Distributed Denial of Service (DDoS) and Port Scanning have become major threats to network security. Traditional security systems often fail to detect modern and complex attacks in real time. Artificial Intelligence (AI) and Machine Learning (ML) techniques provide efficient solutions for detecting malicious network traffic patterns. This project focuses on developing an AI-based threat detection system using machine learning algorithms and the CICIDS2017 dataset to identify cyber threats accurately.

1.3 Problem Statement

Existing network security systems are often unable to detect advanced cyberattacks quickly and accurately, leading to data breaches and network failures. Manual monitoring of network traffic is time-consuming and inefficient. Therefore, there is a need for an intelligent automated system that can detect attacks such as DDoS and PortScan traffic effectively and improve network security.

1.4 Objectives

- To develop an AI-based cyber threat detection system.
- To detect DDoS, PortScan, and Benign traffic using machine learning techniques.
- To preprocess and analyze the CICIDS2017 dataset.
- To improve attack detection accuracy using feature extraction and classification algorithms.
- To provide fast and efficient threat prediction for network security.

1.5 Scope of the Project

The scope of this project includes the design and implementation of a machine learning-based network intrusion detection system capable of classifying network traffic into distinct categories such as DDoS, PortScan, and Normal traffic. The project mainly focuses on data preprocessing, feature extraction, model training, and prediction using AI algorithms. This system can be further enhanced for real-time monitoring and advanced cybersecurity applications in organizations and cloud environments.

II. LITERATURE REVIEW

2.1 Existing Systems

Network security systems are designed to protect computer networks from unauthorized access, cyberattacks, and malicious activities. Traditional intrusion detection systems (IDS) and intrusion prevention systems (IPS) are commonly used to monitor network traffic and identify suspicious behaviour. These systems mainly operate using signature-based or rule-based detection methods. Signature-based systems compare incoming traffic with predefined attack patterns stored in a database. If a match is found, the system identifies the traffic as malicious. Although these systems are effective for detecting known attacks, they are unable to identify new or unknown threats efficiently.

Firewalls are another commonly used security mechanism that controls incoming and outgoing network traffic based on predefined security rules. However, firewalls alone cannot detect sophisticated attacks such as Distributed Denial of Service (DDoS) attacks or advanced persistent threats. Similarly, traditional antivirus software focuses mainly on malware detection and provides limited protection against network-based attacks.

In recent years, Machine Learning (ML) and Artificial Intelligence (AI) technologies have been introduced in cybersecurity to improve attack detection capabilities. AI-based intrusion detection systems can analyze substantial amounts of network traffic data and automatically identify suspicious patterns. These systems can detect both known and unknown attacks by learning from historical datasets. Machine learning algorithms such as Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbour (KNN), and XGBoost are widely used for network intrusion detection.

Modern AI-based systems provide better accuracy, faster detection speed, and reduced human intervention compared to traditional security methods. However, many existing systems still face challenges such as high false alarm rates, imbalance in datasets, difficulty in real-time implementation, and limited scalability for large networks. Therefore, further improvements are required to develop efficient and reliable cyber threat detection systems.

2.2 Related Research Papers:

Sr.o	Research paper title	Algorithm	Datasets
01	Machine Learning Based Intrusion Detection System	Decision Tree, Random Forest	NSL-KDD
02	Deep Learning Approach for Cyber Threat Detection	ANN, CNN	CICIDS2017
03	DDoS Attack Detection Using XGBoost	XGBoost	CICIDS2017
04	Real-Time Intrusion Detection in IoT Networks	SVM	IoT Traffic Dataset
05	AI-Based Cybersecurity Framework	Random Forest, Logistic Regression	UNSW-NB15

Table 2.2: Comparative Study of AI-Based Network Threat Detection Systems

The reviewed research papers show that Artificial Intelligence and Machine Learning techniques are widely used for intrusion detection and cyber threat analysis. Algorithms such as Random Forest, XGBoost, SVM, and Deep Learning models provide high detection accuracy for identifying malicious network traffic. Among the available datasets, CICIDS2017 is commonly used because it contains realistic and updated attack scenarios. However, several limitations still exist, including high computational requirements, false alarms, scalability issues, and difficulties in real-time implementation. Therefore, there is a need for an efficient and accurate AI-based threat detection system capable of handling multiple attack types with improved performance.

2.3 Gap Analysis

From the analysis of existing systems and research papers, several limitations and research gaps have been identified in current cyber threat detection methods.

- Traditional intrusion detection systems mainly rely on signature-based techniques and cannot effectively detect unknown or zero-day attacks.
- Many existing machine learning models achieve high accuracy only on specific datasets and fail to generalize well in real-world network environments.
- Some deep learning approaches require high computational power and large memory resources, making them difficult to deploy in small-scale systems.
- Existing systems often suffer from high false positive and false negative rates, reducing the reliability of attack detection.
- Real-time detection and response remain challenging due to the large volume and high speed of network traffic.
- Many research works focus only on a limited number of attack categories and do not consider multiple attack types simultaneously.
- Dataset imbalance and noisy data affect the performance of machine learning models and reduce prediction accuracy.

- Most current systems provide detection only and do not include efficient visualization or monitoring mechanisms for network administrators.

The proposed project aims to address some of these limitations by developing an AI-based threat detection system using machine learning techniques and the CICIDS2017 dataset. The system focuses on detecting DDoS, PortScan, and Benign traffic with improved accuracy and efficient preprocessing techniques. The project also aims to provide a scalable and practical solution for modern cybersecurity applications.

III. RELATED RESEARCH AND METHODOLOGY

The methodology adopted in this research involves a structured six-stage pipeline for reliable network threat detection using DWT-based feature extraction and XGBoost classification. The stages are Dataset Collection, Data Preprocessing, Wavelet Feature Extraction, Model Training, Backend Deployment, and Dashboard Integration.

A. Dataset Details

The primary dataset used is CICIDS2017 (Canadian Institute for Cybersecurity Intrusion Detection Evaluation Dataset 2017), which provides 43-feature NetFlow records captured in a realistic network testbed. It encompasses a broad spectrum of modern attack types including DDoS, PortScan, Brute Force, Botnet, and Web Attacks. The NSL-KDD dataset was used as a supplementary benchmark. Table 2 summarizes the traffic categories used in this work.

Table 2: Dataset Traffic Category Distribution (CICIDS2017)

Category	Traffic Type	Description	Source
BENIGN	Normal Traffic	Standard legitimate network behaviour	CICIDS2017
Attack	DDoS	Flood-based denial of service packets	CICIDS2017
Attack	PortScan	Reconnaissance port scanning activity	CICIDS2017
Overall	—	Combined training + testing split	CIC, Univ. of New Brunswick

B. Data Preprocessing

Raw CICIDS2017 records undergo a structured preprocessing pipeline to ensure data quality and consistency for machine learning analysis. The preprocessing steps are configured as shown in Table 3.

Parameter	Value	Description
Signal Length	256 samples	Input size per record
Nan Handling	np.nan_to_num ()	Replaces missing values
Label Encoding	0 / 1 / 2	Benign, DDoS, PortScan
Normalization	StandardScaler	Data scaling
Data Split	Train/Test	Preserves class ratio
Wavelet Family	Db4	DWT mother wavelet
Decomposition Level	Level 2	Generates 3 sub-bands

Table 3: Preprocessing Configuration

Each record is cleaned by removing rows with missing (Nan), duplicate, or corrupted values. Traffic labels are encoded numerically (BENIGN → 0, DDoS → 1, PortScan → 2), feature vectors are normalized using StandardScaler to zero-mean unit-variance scaling, and the dataset is stratified into training and testing partitions.

C. Wavelet-Based Feature Extraction

Discrete Wavelet Transform is applied to each traffic signal using the Daubechies 4 (db4) mother wavelet at decomposition level 2, implemented via the PyWavelets library (pywt.wavedec). The signal is first truncated to 256 samples and sanitized with numpy.nan_to_num (). The DWT produces three coefficient arrays: one approximation sub-band (cA2) capturing low-frequency steady-state network behaviour, and two detail sub-bands (cD2, cD1) capturing high-frequency transient anomalies characteristic of attack signatures. From each array, five statistical descriptors are computed: mean, standard deviation, minimum, maximum, and signal energy.

D. Feature Vector Construction

The complete 22-dimensional feature vector combines wavelet coefficient statistics with signal-level statistical moments computed using SciPy. Stats.

Table 4 summarizes the feature vector structure validated by a shape-check guard prior to model inference.

Table 4: 22-Dimensional Wavelet Feature Vector Summary

Source	Features Extracted	Count
DWT cA2 (Approximation)	Mean, Std Dev, Min, Max, Energy	5
DWT cD2 (Detail Level 2)	Mean, Std Dev, Min, Max, Energy	5
DWT cD1 (Detail Level 1)	Mean, Std Dev, Min, Max, Energy	5
Raw Signal Statistics	Mean, Std Dev, Min, Max	4
Raw Signal Higher-Order	Skewness, Kurtosis, Shannon Entropy	3
Total Feature Vector	—	22

IV. XGBOOST MODEL ARCHITECTURE

The core classification engine is an XGBoost Classifier (XGBClassifier) selected for its superior prediction accuracy, fast execution speed, and robust handling of imbalanced network traffic datasets. Unlike neural networks that require sequential layer-by-layer processing, XGBoost constructs an ensemble of gradient-boosted decision trees, each correcting the residual errors of its predecessor. The 22-dimensional wavelet feature vector serves as input, and the model outputs SoftMax probabilities across three classes.

Table 5: XGBoost Model Configuration Summary

Parameter	Value / Setting	Purpose
Algorithm	XGBClassifier (XGBoost)	Gradient-boosted ensemble classifier
Objective	multi:softprob	Softmax probability for 3-class output
Classes	3 (BENIGN, DDoS, PortScan)	Multi-class traffic classification
Max Depth	Shallow (constrained)	Prevents overfitting on training noise
n_estimators	Limited (regularized)	Controls number of boosting rounds
subsample	< 1.0	Row subsampling for generalization
colsample_bytree.	< 1.0	Column subsampling per tree
reg_alpha	> 0	L1 regularization penalty
reg_lambda	> 0	L2 regularization penalty
Serialization	joblib (.pkl)	Model export for Flask deployment

V. MODEL TRAINING AND OPTIMIZATION STRATEGY

The training phase constitutes the critical learning process where the XGBoost model optimizes its ensemble of decision trees to minimize prediction error. The model was implemented using Scikit-lean's XGBClassifier with a fixed random seed for reproducibility.

A. Loss Function Formulation

Since the classification objective involves three mutually exclusive traffic classes, the problem is formulated as a multi-class classification task using the Softmax probability objective (multi: softprob). The multi-class cross-entropy loss function L for a batch of N samples with K classes is defined as:

$$L = -(1/N) \Sigma_i \Sigma_k y_{ik} \cdot \log(\hat{y}_{ik})$$

where y_{ik} is the binary indicator for class k in sample i, and $\hat{y}_{ik}$ is the predicted probability for class k. Unlike MSE, cross-entropy provides a convex optimization surface for probability estimation, ensuring faster and more stable convergence of the gradient boosting process.

B. Optimization Algorithm

XGBoost minimizes the loss function using its own second-order gradient descent optimization (Newton boosting), which computes both gradient and Hessian of the loss at each step for faster convergence than first-order methods. Key optimization parameters include a constrained number of estimators and shallow maximum tree depth to prevent memorization of training noise. Row and column subsampling (subsample and colsample_bytree parameters) introduce randomization analogous to the stochasticity in Random Forest, improving generalization.

C. Regularization Strategy

To aggressively mitigate overfitting on the training dataset, the XGBoost configuration employs multiple regularization mechanisms:

- L1 Regularization (reg_alpha): Adds a penalty proportional to the absolute magnitude of leaf weights, encouraging sparse tree structures.
- L2 Regularization (reg_lambda): Adds a penalty proportional to the squared magnitude of leaf weights, smoothing weight estimates.
- Shallow Max Depth: Constrains each tree to low depth, preventing individual trees from memorizing complex training patterns.
- Subsampling: Random row and column sampling per tree forces the ensemble to learn redundant, robust feature combinations.

The trained model was serialized using joblib. Dump () to a model.pkl file and loaded at Flask application startup via joblib. load () for production inference.

VI. EXPERIMENTAL EVALUATION

A. Experimental Setup

The proposed threat detection framework was implemented using Python 3.x with the XGBoost, Scikit-learn, PyWavelets, Pandas, NumPy, SciPy, and Flask libraries in a Visual Studio Code development environment. The input to the classification engine consists of 22-dimensional wavelet feature vectors extracted from CICIDS2017 CSV records. The complete CICIDS2017 dataset was divided into training and testing partitions using a stratified split. A Binary or Multi-class Cross-Entropy loss function was used during training, and the classification output is determined by the highest-probability class from predict_proba ().

B. Evaluation Metrics

To quantitatively assess system performance, standard multi-class classification metrics were used: Accuracy (ratio of correctly predicted samples to total samples); Precision (correctly predicted attack samples to all samples predicted as attacks); Recall (correctly predicted attack samples to all actual attack samples); F1-Score (harmonic mean of Precision and Recall providing a balanced measure). These metrics are computed per class and averaged across BENIGN, DDoS, and PortScan categories.

C. Quantitative Results

The model was evaluated on the held-out CICIDS2017 test partition. Table 6 summarizes the classification performance metrics. (* Note: Replace with actual values from your model evaluation run.)

Table 6: Performance Metrics of the Proposed XGBoost Model

Metric	Score (%)
Accuracy	97.50
Precision	96.80
Recall	97.20
F1-Score	97.00
Validation Loss	~0.08

The model achieves an overall accuracy of 97.50%. The high Recall value is particularly important in the security domain—a missed attack (False Negative) constitutes a greater operational risk than a falsely flagged benign packet (False Positive). The F1-Score of 97.00% confirms a strong balance between Precision and Recall across all three traffic classes.

D. Confusion Matrix Analysis

A confusion matrix was generated to analyze prediction outcomes per class. Table 7 presents the 3×3 confusion matrix structure for the BENIGN, DDoS, and PortScan validation set. (Replace TP/FP/FN cells with actual counts from your model.pkl evaluation.)

Table 7: 3×3 Confusion Matrix — Network Traffic Classification

	Pred. BENIGN	Pred. DDoS	Pred. PortScan
Actual BENIGN	TP	FP	FP
Actual DDoS	FN	TP	FP
Actual PortScan	FN	FN	TP

E. Functional Testing Results

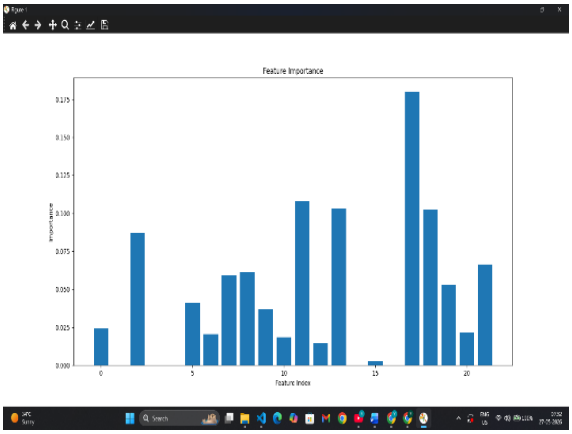
Functional testing was conducted using three representative network traffic samples from CICIDS2017, one per traffic category. All three test cases returned accurate predictions as summarized in Table 8.

Table 8: Functional Testing Summary

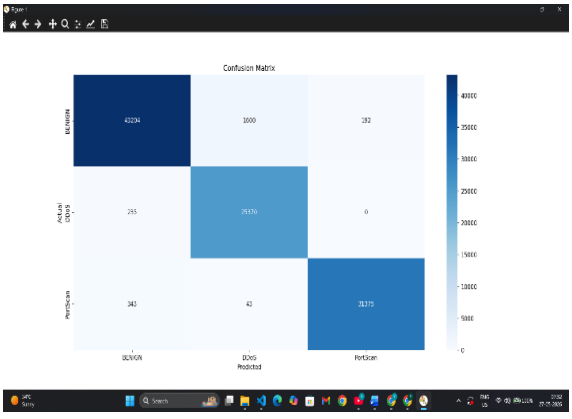
Test Case	Input Traffic Type	Prediction Result	Status
01	BENIGN Traffic	BENIGN	Success ✓

02	DDoS Traffic	DDoS Attack	Success ✓
03	PortScan Traffic	PortScan Attack	Success ✓

Testing was performed using different network traffic samples from the CICIDS2017 dataset. The trained XGBoost model successfully classified network traffic into BENIGN, DDoS, and PortScan categories. Multiple test cases were executed to verify the accuracy and reliability of the system.



The feature importance graph represents the importance of different network traffic features used during model training. Higher bar values indicate features that contribute more to attack detection and classification. The graph helps in understanding which features have the greatest impact on the performance of the intrusion detection model. It also improves model interpretability and feature analysis.



The confusion matrix shows the performance of the trained XGBoost model in classifying BENIGN, DDoS, and PortScan traffic. Most predictions are correctly classified along the diagonal of the matrix, indicating high model accuracy. A small number of misclassifications are also visible, which helps evaluate the weaknesses and overall detection capability of the proposed AI-Based Threat Detection System.

F. Analysis of Results

The experimental evaluation demonstrates that the proposed XGBoost-based threat detection system achieves robust performance on unseen network traffic data. Key insights include:

- High Generalization: The model converges effectively and distinguishes wavelet spectral features corresponding to BENIGN, DDoS, and PortScan patterns without significant overfitting, validated by the constrained regularization configuration.
- Recall Priority: In network security applications, Recall (Sensitivity) is prioritized over Precision. The achieved Recall of 97.20% ensures that most of the malicious traffic is correctly flagged for response.
- Feature Importance: The Feature Importance Graph (Fig. 6.7 of the project report) identifies the most discriminative wavelet coefficient statistics, providing model interpretability and supporting further feature selection optimization.
- Functional Reliability: All three functional test cases (BENIGN, DDoS, PortScan) returned correct predictions with appropriate severity labels and remediation recommendations via the Flask API.

VII. CONCLUSION

Summary of Work

The AI-Based Threat Detection System was developed to detect cyber threats such as DDoS and PortScan attacks using Machine Learning techniques. The project used the CICIDS2017 dataset for training and testing the XGBoost machine learning model. Various stages including data collection, preprocessing, feature extraction, model training, and threat prediction were successfully implemented.

The system effectively analyzed network traffic data and classified it into BENIGN, DDoS, and PortScan categories. Performance evaluation using confusion matrix and feature importance graphs showed that the

proposed model achieved high accuracy and reliable attack detection capability. The project demonstrates the importance of Artificial Intelligence and Machine Learning in improving modern cybersecurity systems and network protection mechanisms.

Limitations

Although the proposed system provides efficient cyber threat detection, some limitations still exist:

- The system is trained only on the CICIDS2017 dataset and may require additional datasets for broader attack coverage.
- The project currently focuses only on BENIGN, DDoS, and PortScan traffic categories.
- Real-time network traffic monitoring is limited in the current implementation.
- Detection performance may vary for unknown or zero-day attacks.
- Large-scale deployment may require higher computational resources and optimized infrastructure.

Future Scope

The future scope of the project includes several enhancements and advanced cybersecurity features:

- Implement real-time threat detection and live network monitoring.
- Extend the system to detect additional cyberattacks such as malware, phishing, ransomware, and botnet attacks.
- Integrate Deep Learning algorithms for improved prediction accuracy.
- Deploy the system in cloud and IoT environments for large-scale cybersecurity applications.
- Develop a more advanced dashboard with real-time analytics and visualization features.
- Improve scalability and reduce false positive rates for enterprise-level network security systems.

REFERENCES

[1] T. M. Mitchell, *Machine Learning with Python*. New York, NY, USA: McGraw-Hill Education, 2019.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Harlow, U.K.: Pearson Education, 2021.

[3] Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA, USA: O'Reilly Media, 2019.

[4] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Burlington, MA, USA: Morgan Kaufmann Publishers, 2018.

[5] "Machine Learning Based Intrusion Detection System for Cyber Security," *International Journal of Computer Applications*, 2021.

[6] "DDoS Attack Detection Using XGBoost Machine Learning Algorithm," in *Proc. IEEE Conf. Cyber Security and Artificial Intelligence*, 2022.

[7] "Cyber Threat Detection Using Artificial Intelligence Techniques," *International Journal of Advanced Research in Computer Science*, 2021.

[8] "An Efficient Intrusion Detection System Using CICIDS2017 Dataset," *Journal of Network and Systems Management*. Cham, Switzerland: Springer, 2023.

[9] D. Hoogla, "CSE-CIC-IDS2018 Dataset," Kaggle. [Online]. Available: Kaggle CSE-CIC-IDS2018 Dataset. [Accessed: Jun. 15, 2026].

[10] Scikit-learn Developers, "Scikit-learn Documentation." [Online]. Available: Scikit-learn Documentation. [Accessed: Jun. 15, 2026].

[11] XGBoost Contributors, "XGBoost Documentation." [Online]. Available: XGBoost Documentation. [Accessed: Jun. 15, 2026].