AI-Powered Intelligent Framework For Detection And Prevention Of Cybersecurity Attacks Using Machine Learning Algorithms

Giritharan R[1], Dr. K. Annalakshmi[2]
[1]Dept of Computer Applications
[2]Assistant professor, Dept of MBA
[1, 2] Dr. M.G.R. Educational and Research Institute
Deemed to be University, Chennai, Tamil Nadu, India

Abstract- *The rapid growth of digital technologies has substantially increased exposure to cybersecurity threats including malware, phishing, ransomware, and unauthorized network intrusions. Traditional signature-based security systems are inherently reactive and fail against zero-day exploits and polymorphic attacks. This paper presents ThreatGuardian, an AI-powered cybersecurity threat detection and prevention framework that leverages machine learning algorithms — Random Forest (RF), AdaBoost Classifier (ADC), and Bernoulli Naive Bayes Classifier (BNC) — to identify and classify malicious network activities in real time. The system integrates data preprocessing, exploratory data analysis, feature extraction, and model evaluation pipelines with a Django-based web interface for practical deployment. Trained and evaluated on a publicly available network intrusion dataset from Kaggle, the best-performing model is serialized and deployed for real-time inference. Performance evaluation using accuracy, precision, recall, and F1-score demonstrates that the proposed framework significantly outperforms traditional rule-based methods, providing an adaptive, scalable, and user-accessible solution for modern cyber threat management.*

Keywords: Cybersecurity, Machine Learning, Intrusion Detection System, Random Forest, AdaBoost, Bernoulli Naive Bayes, Network Threat Detection, Django, Real-Time Monitoring.

I. INTRODUCTION

The proliferation of internet-connected devices and cloud-based services has dramatically expanded the attack surface of digital infrastructures. Cybersecurity threats such as Distributed Denial of Service (DDoS), ARP poisoning, brute-force attacks, and ransomware continue to grow in both sophistication and frequency. Traditional Intrusion Detection Systems (IDS) rely on static rule-sets and signature databases, rendering them ineffective against novel or zero-day exploits that lack prior signatures [1].

Artificial Intelligence (AI) and Machine Learning (ML) have emerged as transformative technologies in cybersecurity. Unlike rule-based systems, ML models can learn behavioral patterns from historical data and generalize to detect previously unseen attack vectors. This capacity for adaptive learning enables proactive threat identification rather than purely reactive defense.

This paper proposes ThreatGuardian, an AI-driven cybersecurity framework that combines multiple supervised learning algorithms with a full-stack web deployment to classify network traffic as benign or malicious across twelve distinct attack categories. The primary contributions of this work are: (1) a comparative evaluation of Random Forest, AdaBoost, and Bernoulli Naive Bayes classifiers for network threat classification; (2) integration of the optimal model into a Django web application enabling accessible real-time prediction; and (3) a comprehensive data preprocessing and visualization pipeline tailored to high-dimensional network flow data.

II. RELATED WORK

Numerous studies have explored ML for network intrusion detection. Lad and Adamuthe [2] demonstrated improved malware classification accuracy using convolutional neural network architectures on binary visualizations of executable files. Zhong and Cheng [3] proposed adversarial malware generation using Conv-GANs to stress-test black-box detectors, highlighting vulnerabilities in static classifiers.

Ensemble methods have received particular attention due to their robustness. Freund and Schapire [4] introduced AdaBoost, which iteratively combines weak learners into a strong classifier by up-weighting misclassified instances. Subsequent work has applied AdaBoost variants to network anomaly detection with favorable results. Random Forest, introduced by Breiman, constructs multiple decorrelated decision trees and aggregates their predictions, offering

resistance to overfitting and built-in feature importance ranking [5].

Despite these advances, few systems integrate high-accuracy ML backends with accessible, non-expert user interfaces. Most tools require command-line interaction or expert configuration, limiting practical adoption. ThreatGuardian addresses this gap by deploying the trained classifier through a Django web interface accessible to non-technical users.

III. SYSTEM ARCHITECTURE AND DESIGN

The ThreatGuardian framework comprises five interconnected layers: data acquisition, preprocessing, model training and selection, deployment, and the user interface.

A. Data Acquisition

The dataset was sourced from Kaggle and contains network traffic records labeled across twelve attack categories: ARP Poisoning, DDOS Slowloris, DOS SYN Hping, Metasploit Brute Force SSH, MQTT Publish, NMAP FIN Scan, NMAP OS Detection, NMAP TCP Scan, NMAP UDP Scan, NMAP XMAS Tree Scan, ThingSpeak, and Wipro Bulb, alongside benign traffic. Each record contains 46 numerical network flow features including flow duration, forward/backward packet statistics, header sizes, flag counts, and payload statistics.

B. Preprocessing Pipeline

Raw data ingested via Pandas DataFrames undergoes the following pipeline: (i) missing value imputation using column-wise median substitution; (ii) duplicate record removal; (iii) outlier detection using Z-score analysis; (iv) label encoding for categorical variables; and (v) feature scaling via min-max normalization. The processed dataset is split 80:20 for training and testing respectively.

C. Exploratory Data Analysis

Matplotlib and Seaborn are used to generate correlation heatmaps, class distribution bar charts, box plots for outlier visualization, and pair plots for inter-feature relationship analysis. These visualizations informed feature selection and helped identify class imbalances corrected through stratified sampling.

D. Model Training and Selection

Three classifiers are trained and evaluated using scikit-learn: Random Forest (RF), AdaBoost Classifier (ADC), and Bernoulli Naive Bayes Classifier (BNC). K-fold cross-validation (k=5) is employed to estimate generalization performance. The best-performing model by weighted F1-score is serialized using joblib into a .pkl file for deployment.

E. Deployment

The serialized model is integrated into a Django web application. Upon receiving a POST request containing 46 feature values via an HTML form, the view deserializes the model, reshapes the input vector to (1, -1) using NumPy, invokes prediction, maps the integer output to a human-readable attack label, persists the record to SQLite3, and renders a result page with remediation guidance for the identified attack type.

IV. ALGORITHMS

A. Random Forest

Random Forest [5] is an ensemble method that builds N decision trees, each trained on a bootstrap sample of the training data with a random feature subset considered at each split. Classification is by majority vote. This reduces variance relative to individual trees while maintaining low bias. Key hyperparameters include the number of estimators, maximum tree depth, and minimum samples per leaf. RF also provides a feature importance score via mean decrease in Gini impurity, enabling post-hoc interpretability.

B. AdaBoost Classifier

AdaBoost [4] is a boosting ensemble method that trains a sequence of weak learners (typically depth-1 decision stumps) on iteratively re-weighted training samples. Misclassified instances receive higher weights, directing subsequent learners toward difficult cases. The final prediction is a weighted majority vote of all weak learners. AdaBoost is resistant to overfitting in low-noise environments and adapts effectively to complex decision boundaries.

C. Bernoulli Naive Bayes

The Bernoulli Naive Bayes (BNC) classifier models each feature as a binary variable and applies Bayes' theorem under the naive independence assumption. Despite this simplifying assumption, BNC achieves competitive performance on high-dimensional sparse data and offers extremely fast training and inference. Laplace smoothing is applied to handle zero-probability features.

V. IMPLEMENTATION

A. Technology Stack

Table I. Technology Stack

Component	Technology
Language	Python 3.8+
ML Library	scikit-learn
Data Processing	NumPy, Pandas
Visualization	Matplotlib, Seaborn
Web Framework	Django
Database	SQLite3
Model Serialization	joblib (.pkl)
IDE	Anaconda / Jupyter

B. Module Workflow

The system workflow proceeds as follows: (1) Import library packages and read the Kaggle network threat dataset; (2) Perform data validation, cleaning, and preprocessing; (3) Conduct exploratory data analysis and visualization; (4) Split data 80:20 and train RF, ADC, and BNC classifiers; (5) Compare models on the test set and select the best by weighted F1-score; (6) Serialize the winning model to CYBER.pkl; (7) Integrate the model into Django views; (8) Accept user input via the web form (46 features), perform prediction, store to SQLite3, and display results.

C. Django Prediction View

The core prediction logic loads the serialized model once at application startup using joblib.load(). On form submission, feature values are extracted from the POST payload, assembled into a NumPy array, reshaped to (1, -1), and passed to Model.predict(). The integer prediction index is mapped to one of twelve attack label strings (e.g., prediction==2 maps to 'DOS_SYN_Hping'). The result, along with the full feature set, is persisted as a UserPredictModel instance in SQLite3 and rendered to the output template.

VI. RESULTS AND EVALUATION

A. Performance Metrics

Model performance is evaluated using four standard classification metrics: Accuracy = (TP + TN) / (TP + TN + FP + FN), Precision = TP / (TP + FP), Recall = TP / (TP + FN), and F1-Score = 2 × (Precision × Recall) / (Precision + Recall). These metrics together provide a holistic view of classifier performance, particularly important for multi-class problems with potential class imbalance.

Table II. Comparative Model Performance

Algorithm	Accuracy	Precision	Recall	F1-Score
Random Forest	98.7%	98.5%	98.6%	98.5%
AdaBoost	96.4%	96.1%	96.2%	96.1%
Bernoulli NB	89.3%	89.0%	88.8%	88.9%

B. Discussion

Random Forest achieved the highest accuracy of 98.7%, attributed to its ensemble structure and ability to model non-linear decision boundaries across 46 features. AdaBoost performed competitively at 96.4%, demonstrating the effectiveness of iterative boosting for imbalanced classes. Bernoulli Naive Bayes, while simpler and faster, yielded 89.3% accuracy due to the independence assumption not fully holding for correlated network flow features.

All test cases in the test report passed successfully, validating the complete pipeline from data loading through Django deployment. The system correctly classified all twelve attack categories including ARP Poisoning, DDOS Slowloris, and Metasploit Brute Force SSH.

C. Test Report Summary

Twelve functional test cases covering dataset loading, preprocessing, model training for all three algorithms, model comparison, attack prediction, Django deployment, and result display were all executed and passed. Authentication tests (TC_AUTH_01 through TC_AUTH_05) confirmed secure session management and password reset functionality.

VII. CONCLUSION AND FUTURE WORK

This paper presented ThreatGuardian, an end-to-end AI-powered cybersecurity threat detection framework combining Random Forest, AdaBoost, and Bernoulli Naive Bayes classifiers with a Django web application. The system addresses the core limitations of traditional rule-based IDS: inability to detect novel attacks, high false positive rates, and inaccessibility to non-expert users. Random Forest achieved the best performance at 98.7% accuracy across twelve network attack categories.

The Django deployment bridges the gap between ML research and practical cybersecurity operations, enabling real-time threat prediction through an intuitive web interface without requiring command-line expertise. The SQLite3

persistence layer maintains a historical log of predictions for audit and trend analysis.

Future work will focus on: (1) integrating deep learning models (LSTM, CNN) for sequential network traffic analysis; (2) connecting to live network interfaces for real-time packet capture and streaming inference; (3) cloud-based deployment on AWS or Azure for scalability; (4) blockchain-based immutable logging of threat events; and (5) federated learning to enable collaborative model improvement without centralizing sensitive network data.

REFERENCES

[1] W. Stallings and L. Brown, Computer Security: Principles and Practice, 4th ed. Pearson Education, 2018.
[2] S. S. Lad and A. C. Adamuthe, "Malware Classification with Improved Convolutional Neural Network Model," International Journal of Computer Applications, 2020.
[3] F. Zhong and X. Cheng, "MalFox: Camouflaged Adversarial Malware Example Generation Based on Conv-GANs Against Black-Box Detectors," IEEE Transactions on Information Forensics and Security, 2023.
[4] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119–139, 1997.
[5] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[6] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd ed. O'Reilly Media, 2022.
[7] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.
[8] Scikit-Learn Developers, Scikit-Learn User Guide. [Online]. Available: https://scikit-learn.org