International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 10, Issue 4, April-2025, http://ijmec.com/, ISSN: 2456-4265

Machine Learning-Driven Intrusion Detection: Enhancing Security Against Brute Force Attacks

Mr. Mohammed Mazheruddin [*1], Mr. Syed Arshil Hussain [*2], Mr. Junaid Mohiuddin [*3], Mr. Syed Fawaz Ur Rahman [*4],

[*1] Assistant Professor, Dept. of CSE-AIML, Lords Institute of Engineering and Technology

[*2, 3, 4] B.E Student Dept. of CSE-AIML, Lords Institute of Engineering and Technology

mdmazher@lords.ac.in[*1], syedarshilhussain@gmail.com[*2], junaidmohiuddin7hyd@gmail.com*[3], notfawaz.2100@gmail.com[*4]

ABSTRACT

With the rise in internet usage, networks face increasing cyber threats, particularly brute force attacks on services like SSH and FTP. The project "Castpone" aims to develop a Machine Learning-based Intrusion Detection System (IDS) to efficiently detect such attacks. Utilizing the CSE-CIC-IDS2018 dataset and a custom data capture setup, the approach involves traffic acquisition, feature extraction via CICFlowMeter, preprocessing, and intelligent feature selection. Multiple classifiers—Decision Tree, Random Forest, Naïve Bayes, K-Nearest Neighbors, and Multi-Layer Perceptron—were evaluated using metrics like Precision, Recall, F1-Score, Accuracy, and computational efficiency. Decision Tree and Random Forest models outperformed others, proving effective for real-time intrusion detection due to their speed and accuracy.

The study underscores the importance of model selection and feature engineering, with Random Forest-based feature selection reducing overhead without sacrificing performance. Overall, "Castpone" lays the groundwork for scalable, adaptive IDS solutions suitable for evolving cybersecurity landscapes.

1. INTRODUCTION

1.1 GENERAL

As cyber threats grow in frequency and complexity, traditional Intrusion Detection Systems (IDS) struggle to keep up. This section highlights the transition to Machine Learning (ML)-based IDS, which can learn from historical data and adapt to new, dynamic attack patterns—especially brute force attacks. With increasing reliance on digital networks for sensitive operations, intelligent and adaptive security frameworks have become essential.

1.2 PROJECT OVERVIEW

The project "Castpone" focuses on developing a ML-based IDS to detect brute force attacks effectively. Using the CSE-CIC-IDS2018 dataset and real-time traffic capture tools, the project involves flow-level feature extraction via CICFlowMeter, followed by data preprocessing and normalization. Multiple classifiers—Decision Tree, Random Forest, Naïve Bayes, KNN, and MLP—are trained and evaluated using metrics such as Accuracy, Precision, Recall, and F1-Score. Visual tools aid in comparing model performance to identify the most efficient and reliable algorithms for real-time application.

1.3 OBJECTIVE

- Develop a robust ML-based IDS for brute force attack detection.
- Explore data preprocessing and feature engineering impacts on model performance.

- Compare classifiers to identify optimal accuracy and efficiency trade-offs.
- Offer insights for real-world deployment of scalable and adaptive IDS solutions.

2. LITERATURE SURVEY

1. TOWARD GENERATING A NEW INTRUSION DETECTION DATASET AND INTRUSION TRAFFIC CHARACTERIZATION (2018)

Authors: Sharaf Aldin, Iman; Lashkari, Arash H.; Ghorbani, Ali

Introduces the CICIDS2017 dataset with realistic, labeled traffic and modern attack types, ideal for ML-based IDS training and evaluation.

2. TOWARDS THE DEVELOPMENT OF REALISTIC BOTNET DATASET IN THE INTERNET OF THINGS FOR NETWORK FORENSIC ANALYTICS: BOT-IOT (2018)

Authors: Koroniotis, Nickolaos; Moustafa, Nour; Sitnikova, Elena

Proposes Bot-IoT dataset simulating real IoT traffic and attacks like DDoS and data theft, offering over 70M labeled records for ML/forensics.

3. A SURVEY OF NETWORK-BASED INTRUSION DETECTION DATASETS (2019)

Authors: Ring, Markus; Wunderlich, Steffen; et al.

Compares datasets (KDD99, CICIDS2017, etc.) based on realism and diversity. Highlights issues like outdated attacks and class imbalance, calling for standardized, modern datasets.

4. TOWARDS A RELIABLE COMPARISON AND EVALUATION OF NETWORK IDS BASED ON ML (2020)

Authors: Ferrag, Mohamed A.; Maglaras, Leandros

Addresses inconsistencies in IDS evaluation. Proposes a benchmarking framework for fair ML model comparisons using standardized datasets and metrics.

5. OPTIMIZING IDS IN THREE PHASES ON THE CSE-CIC-IDS2018 DATASET (2023)

Authors: Farzaneh, Mostafa; Javadi, Bahman

Uses a three-phase approach—preprocessing, feature selection, and classification—to boost IDS performance using models like Random Forest and XGBoost.

6. LITNET-2020: AN ANNOTATED REAL-WORLD NETWORK FLOW DATASET (2020)

Authors: Paulauskas, Nerijus; Garsva, Gintautas

Presents LITNET-2020 with real-world academic traffic and labeled attacks. Its authenticity makes it ideal for developing practical ML-based IDS.

7. A COMPREHENSIVE SURVEY OF NETWORK ANOMALY DETECTION (2016)

Authors: Chandola, Varun; Banerjee, Arindam; Kumar, Vipin

Reviews anomaly detection methods—statistical, ML, and spectral. Discusses real-time challenges, encrypted traffic, and benchmarking limitations.

8. SSH-BRUTEFORCE ATTACK DETECTION MODEL BASED ON DEEP LEARNING (2021)

Authors: Sangkatsanee, Poom; et al.

Develops an LSTM-based model to detect SSH brute force attacks. Captures time-series login patterns, outperforming traditional ML in accuracy.

9. A DEEP LEARNING APPROACH FOR IDS (2019)

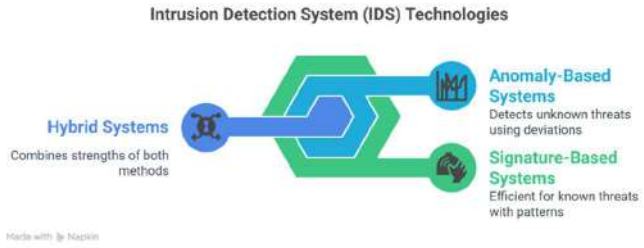
Authors: Kim, Jaehyun; Kim, Heejo

Utilizes DNNs for automatic feature learning and improved detection of unseen attacks, reducing false positives over legacy classifiers.

10. DETECTING DDoS ATTACKS USING DATA MINING TECHNIQUES
Authors: Mirkovic, Jelena; Reiher, Peter

Applies classification and clustering for DDoS detection. Identifies anomalies in traffic patterns and improves real-time detection accuracy.

3. SYSTEM ANALYSIS



3.

1 EXISTING SYSTEM

Intrusion Detection Systems (IDS) are traditionally classified into:

Signature-Based IDS: Detect known threats using patterns. Efficient but ineffective against new attacks.

Anomaly-Based IDS: Use ML/statistical models to identify deviations from normal behavior. Better at spotting unknown threats but prone to false positives.

Hybrid IDS: Combine both approaches for broader detection coverage.

Limitations of Existing Systems:

- Poor detection of evolving threats
- Scalability and resource issues
- High false positives/negatives
- Inability to operate in real-time
- Reduced accuracy with subtle or complex attacks

3.2 PROPOSED SYSTEM

The proposed IDS leverages machine learning to detect brute force attacks (e.g., SSH/FTP) with higher accuracy and adaptability.

Key Features:

- Supervised ML models trained on labeled traffic data
- Feature selection using Random Forest importance
- Automated preprocessing (cleaning, encoding, normalization)
- Comparative evaluation of classifiers (DT, RF, KNN, MLP, NB)
- Realistic dataset based on simulated attacks (CSE-CIC-IDS2018 format)

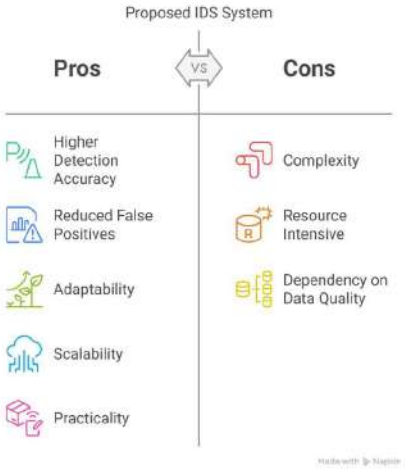
Workflow:

1. Simulated brute force attacks on CentOS VM (SSH/FTP)
2. Traffic capture and flow generation using CIC FlowMeter
3. Preprocessing and feature selection
4. Model training and evaluation using performance metrics

3.2.1 ADVANTAGES

- High Accuracy in detecting known and unknown attacks
- Reduced False Positives/Negatives via feature optimization
- Adaptability to evolving threats through retraining
- Scalability using cloud tools like Google Colab

- Real-world Applicability with open-source tools and realistic
- traffic data



4. REQUIREMENT SPECIFICATIONS

4.1 SOFTWARE REQUIREMENTS

- Language: Python – for ML and data science support
- Platform: Google Colab – free cloud-based GPU/CPU access
- Libraries:
- Data Handling: NumPy, Pandas
- ML: Scikit-learn
- Visualization: Matplotlib, Seaborn
- Traffic Analysis: CIC FlowMeter – for generating flow-based traffic data
- Purpose: To enable efficient development, processing, and evaluation of the IDS system

4.2 HARDWARE REQUIREMENTS

- Victim Machine: CentOS 7 VM (2 GB RAM, 2 vCPUs) with SSH/FTP services
- Attacker Machine: Kali Linux VM (2 GB RAM, 2 vCPUs) using Patator for brute force attacks
- Cloud Resource: Google Colab (Intel Xeon, 2 vCPUs, 13 GB RAM) for scalable ML processing
- Setup: Hybrid environment (VMs + cloud) to simulate real-world attack scenarios
- Purpose: To ensure accurate, controlled IDS testing in a realistic setting

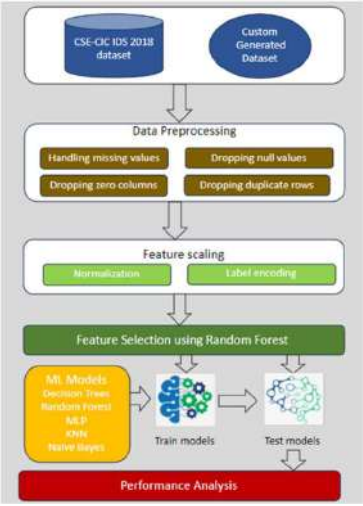
5. SYSTEM DESIGN

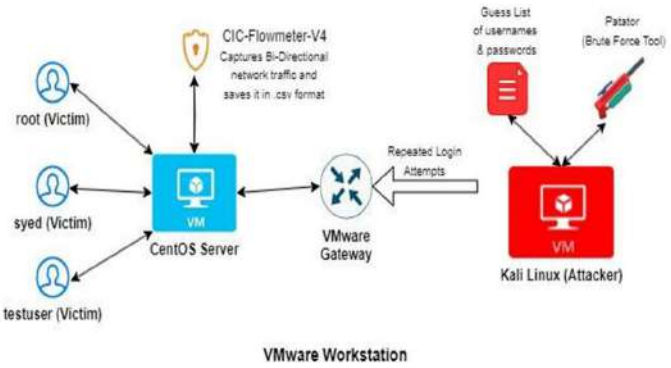
5.1 SYSTEM ARCHITECTURE

The proposed intrusion detection system follows a modular and scalable architecture designed to effectively identify brute force attacks on SSH and FTP services. The process begins with the simulation of realistic attacks using Patator on a CentOS virtual machine, with network traffic captured in .pcap format via Wireshark or tcpdump. This raw packet data is then converted into structured flow-based .csv files using CIC FlowMeter, which summarizes traffic sessions through key statistical features. The resulting data undergoes thorough preprocessing, including missing value handling, outlier removal, encoding of categorical variables, and normalization to ensure consistency. A Random Forest-based feature selection mechanism is then applied to

identify the most significant attributes, improving detection accuracy and reducing computational complexity. Finally, multiple machine learning classifiers such as Decision Tree, Random Forest, K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), and Naïve Bayes are trained and evaluated using metrics like accuracy, precision, recall, and F1-score. The best-performing model is selected for deployment. This structured workflow ensures efficient data handling, accurate detection, and adaptability to evolving cyber threats

.





5.2

UML DIAGRAMS

1. Use Case Diagram – Workflow

Shows how external users (e.g., network admin) interact with system functionalities like starting packet capture, preprocessing data, training the model, and detecting intrusions

2. Class Diagram – Workflow

Represents system structure with classes like PacketSniffer, FlowGenerator, ModelTrainer, etc., including their attributes, methods, and relationships.

3. Object Diagram – Workflow

Displays runtime instances of classes (e.g., sniffer1, flowGenA) and how data flows between them during system execution.

4. Sequence Diagram – Workflow

Depicts the order of operations: packet capture → flow generation → preprocessing → model training → intrusion detection, showing interactions over time.

5. Activity Diagram – Workflow

Illustrates the process flow from packet capture to attack detection, including decision points like "Is data sufficient?"

6. State Diagram – Workflow

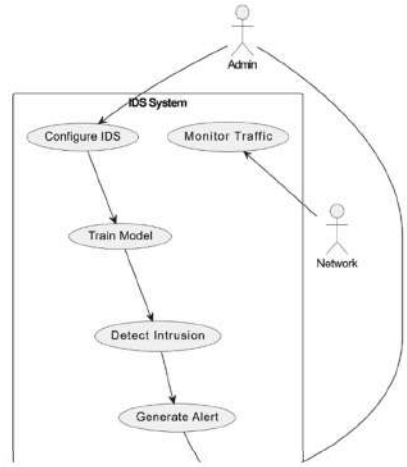
Shows system states like Idle, Capturing, Processing, Detecting, and transitions based on events or actions.

7. Component Diagram – Workflow

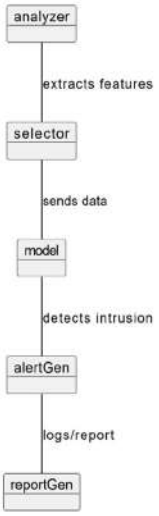
Breaks the system into components: UI, Capture Module, Preprocessing, ML Trainer, Detector, and Logger, showing their interconnections.

8. Deployment Diagram – Workflow

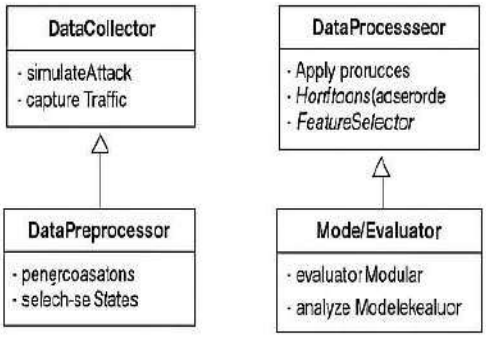
Maps software modules onto hardware (local server, cloud), showing network communication between capture, detection, and user systems.

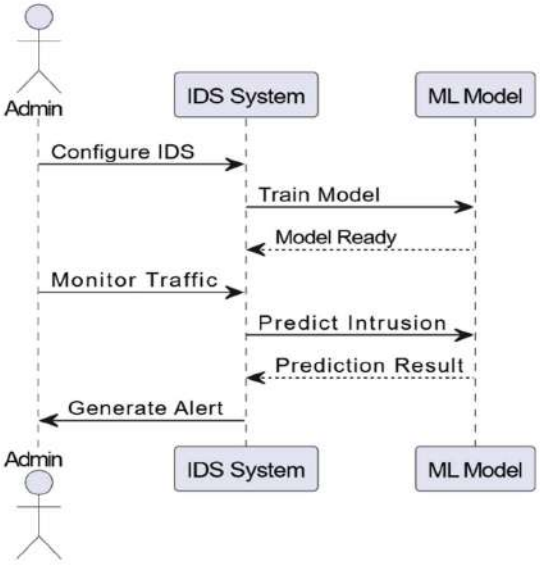


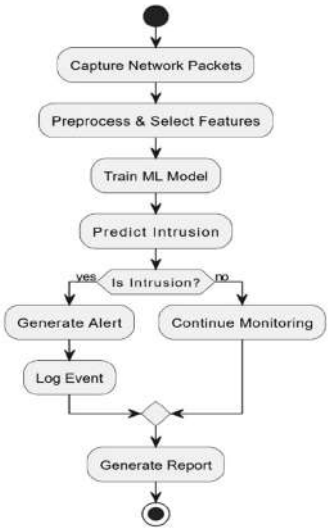
Object Diagram - Intrusion Detection System (Runtime Snapshot)



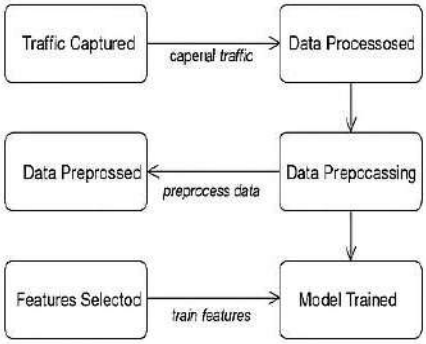
Class Diagram







State Diagram



5.3

MODULES

1. Packet Capture Module

Captures live network traffic using tools like Wireshark or sniffing libraries and forwards it for flow conversion.

2. Flow Generation Module

Converts raw packets into flow-based data using CICFlowMeter, summarizing key features like IPs, duration, bytes, and protocol.

3. Feature Extraction & Preprocessing Module

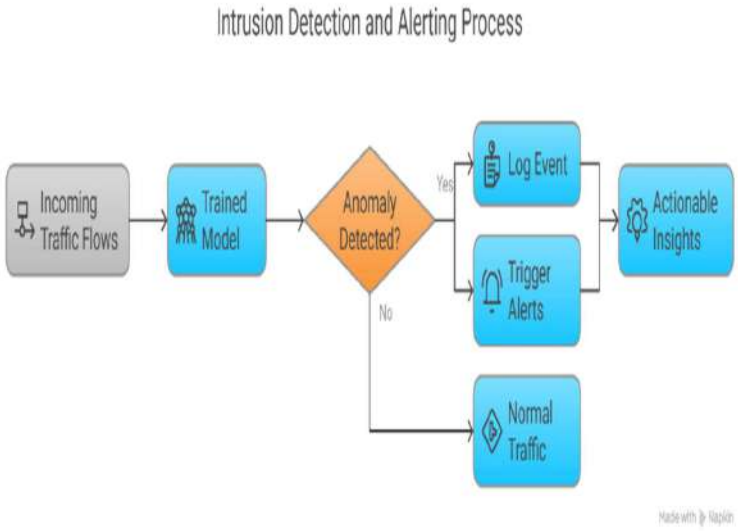
Cleans and transforms the data—removing noise, encoding categorical values, and normalizing features for ML model input.

5. Model Training Module

Trains machine learning models (e.g., Random Forest, Decision Tree) on processed data, evaluating performance via accuracy, precision, recall, and F1-score.

6. Intrusion Detection & Alerting Module

Classifies traffic using the trained model, detects brute-force patterns, logs incidents, and triggers alerts if necessary.



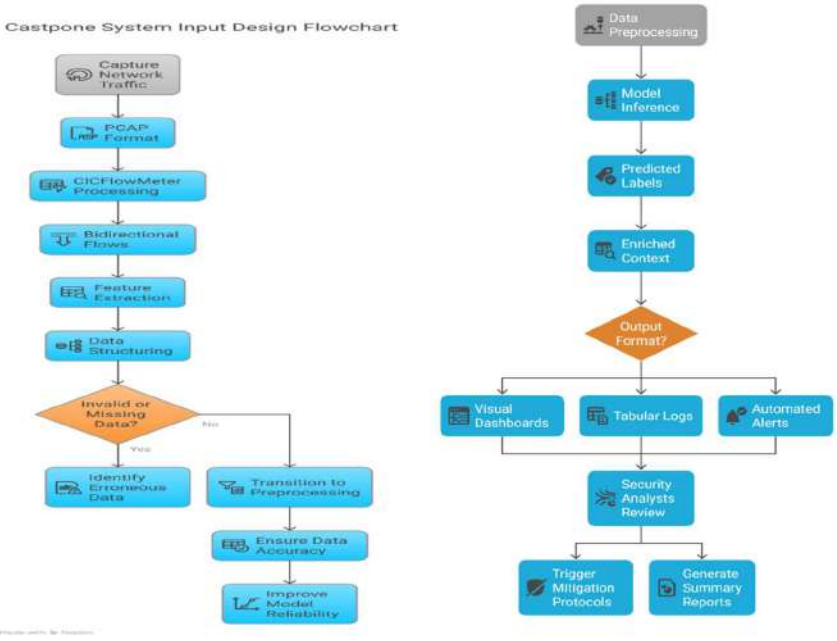
7. IMPLEMENTATION

6.1 INPUT DESIGN

Castpone's input begins with live or stored network traffic in PCAP format. This raw data is converted into structured flow records using CICFlowMeter, extracting features like duration, byte count, protocol, and IPs. The design ensures accurate and relevant data for effective machine learning classification.

6.2 OUTPUT DESIGN

The system outputs classification results (benign or malicious) with details like timestamps, IPs, and confidence scores. In real-time, it can trigger alerts and log suspicious flows, offering clear and actionable insights for administrators.



6.3

SAMPLE CODE

The Castpone system uses modular Python code built on pandas, numpy, scikit-learn, and joblib for model training and prediction. The process includes:

Data Preprocessing:

Cleaning the dataset by removing duplicates, nulls, zero-only columns, and categorical fields (e.g., "Timestamp"). This ensures compatibility with ML models and improves data quality.

Feature Scaling:

Applying normalization to input features and label encoding to output classes for consistent and unbiased model training.

Feature Selection:

Using Random Forest to rank and retain the most important features, reducing complexity and enhancing performance.

Model Training:

Training various classifiers—Decision Tree, Random Forest, MLP, Naive Bayes, and KNN—to detect brute-force attacks. The best-performing model is serialized using joblib.

Prediction:

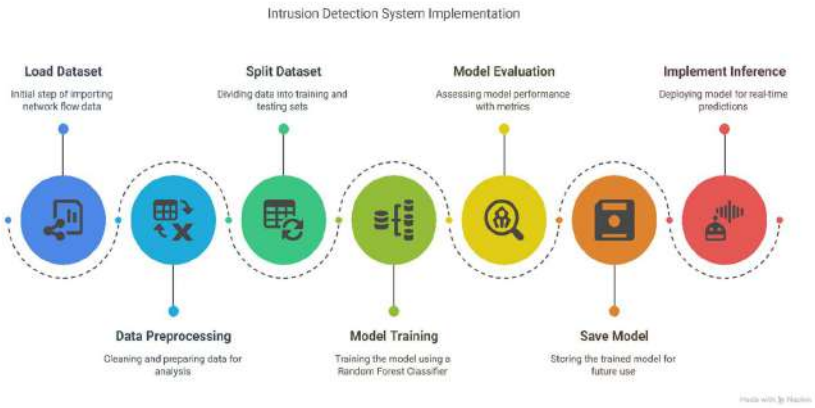
A separate script loads the trained model and classifies new flows in real-time or batch settings.

6.4 IMPLEMENTATION

The implementation phase executes the full intrusion detection pipeline—from preprocessing to prediction. The dataset, containing flow-based network features, is loaded and cleaned by removing irrelevant fields and handling missing values. Categorical labels are encoded for compatibility.

The data is split into training (70%) and testing (30%) sets. A Random Forest Classifier is trained to detect brute force patterns and evaluated using accuracy, precision, recall, and F1-score.

The trained model and scaler are saved using joblib for future predictions. An inference function processes new flows and predicts intrusions, enabling real-time or batch detection. This implementation confirms the system's effectiveness in identifying brute force attacks.



8. SOFTWARE TESTING

Software testing ensures the reliability and correctness of the brute force detection system. It verifies the end-to-end functionality—from data preprocessing to prediction.

Unit Testing was conducted on individual modules like data loaders and preprocessing functions to ensure proper handling of nulls, label encoding, and scaling.

Integration Testing validated the seamless data flow between components, ensuring compatibility in formats and feature consistency.

Model Evaluation involved testing the Random Forest classifier using metrics like accuracy, precision, recall, F1-score, and a confusion matrix to assess classification performance.

Performance Testing measured the system's ability to process large datasets efficiently, confirming suitability for real-time detection scenarios.

The system consistently produced accurate and stable results. Minor issues were optimized, confirming the model's robustness and deployment readiness.

Key Metrics Explained:

Confusion Matrix: Shows TP, TN, FP, FN for evaluating model predictions.

Accuracy: Ratio of correct predictions to total predictions.

Precision: Correct positive predictions over total predicted positives.

Recall: Correct positives over total actual positives.

F1-Score: Harmonic mean of precision and recall for balanced performance.

9. RESULT ANALYSIS

It demonstrated and evaluated multiple machine learning algorithms based on accuracy, speed, and suitability for real-time intrusion detection. Metrics such as Precision, Recall, F1-Score, Accuracy, training time, and prediction latency were used.

Training Time:

- **MLP (Neural Network)**:

Took the longest training time (~10 seconds).

Suitable for static systems, not frequent retraining.

- **Decision Tree, Random Forest, Naive Bayes, KNN**:

Required minimal training time (close to 0 seconds).

Highly efficient for rapid model development and frequent updates.

Prediction Time:

- **K-Nearest Neighbors (KNN)**:

Slowest in prediction (~0.14 seconds).

Not ideal for real-time IDS.

- **Decision Tree & Random Forest**:

Fastest predictions (<0.02 seconds).

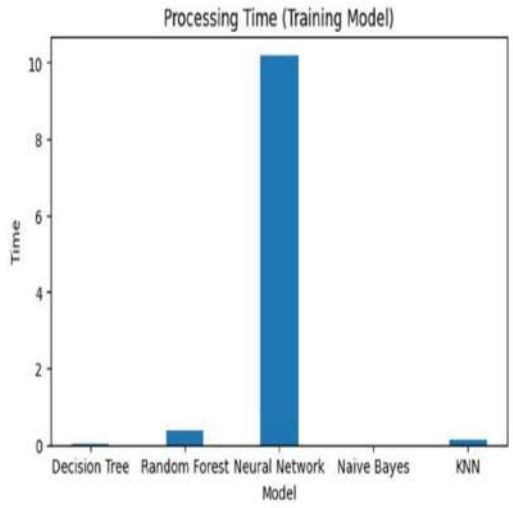
Well-suited for real-time detection.

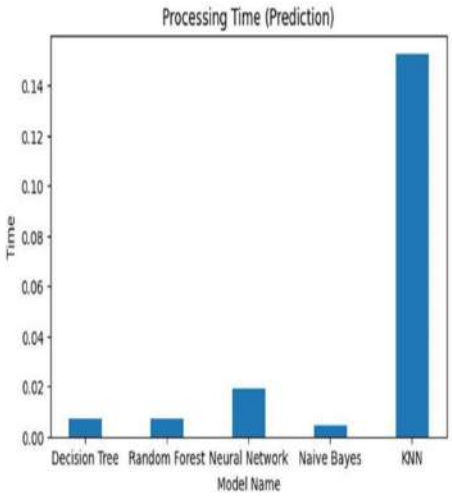
- **MLP & Naive Bayes**:

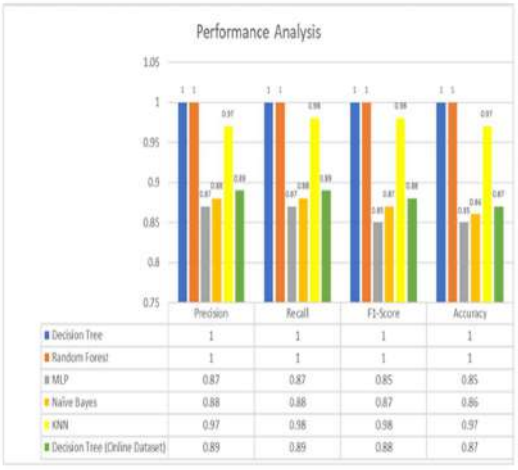
Moderate prediction speed, better than KNN.

Performance Metrics:

- Decision Tree & Random Forest: Perfect scores (1.00) across all metrics.
- MLP: 0.87 for Precision, Recall, Accuracy; F1-score was 0.85.
- Naive Bayes: 0.88 for key metrics; Accuracy at 0.86.
- KNN: Precision 0.97, Recall 0.98, F1-score 0.98, Accuracy 0.97.
- Online Decision Tree: Consistent 0.89 across metrics.

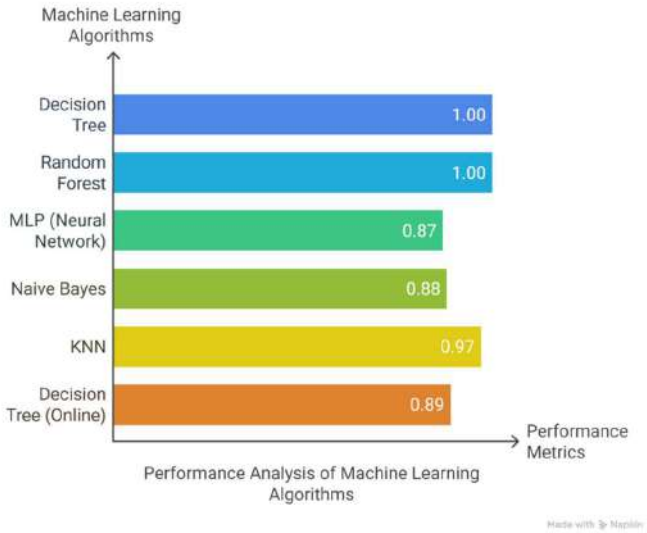






Overall comparison:

Decision Tree and Random Forest models offer the best trade-off between accuracy and efficiency. KNN, though accurate, is slower in prediction. The results support using Decision Tree or Random Forest for scalable, real-time intrusion detection systems.



10. FUTURE SCOPE & CONCLUSION

10.1 FUTURE SCOPE

The current system excels in detecting brute-force attacks but has ample potential for expansion. Future improvements include real-time deployment for continuous traffic monitoring, multi-attack classification to detect diverse network attacks like DDoS and phishing, and integrating advanced deep learning models (CNNs and RNNs) for better pattern recognition and analysis. A cloud-based IDS would enable scalable, distributed monitoring across larger networks, while automated threat responses could alert administrators or take preventive actions in real time. Additionally, research can further explore advanced machine learning techniques, ensemble modeling, and real-time threat intelligence to enhance the adaptability and strength of IDS systems in response to evolving cyber threats.

10.2 CONCLUSION

This study on Intrusion Detection Systems (IDS) emphasizes the importance of machine learning (ML) in cybersecurity, specifically in detecting brute-force attacks. By utilizing the CSE-CIC-IDS-2018 dataset and a customized data collection environment, the research uncovered valuable insights into the efficacy of ML models for intrusion detection. Key findings include the identification of influential features that improve classification precision and reduce false positives and negatives. Among the evaluated models, Decision Tree and Random Forest exhibited exceptional accuracy and efficiency, confirming their suitability for real-time IDS deployment. Feature selection, especially using Random Forest, improved model performance and execution speed. This work lays the foundation for future research into advanced machine learning strategies and scalable IDS solutions, which will be crucial for adapting to the ever-changing landscape of cyber threats.

11. BIBLIOGRAPHY

1. Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset. Link

2. SSH-Brute Force Attack Detection Model based on Deep Learning. ResearchGate, 2023. Link

3. Aljanabi, M., Ismail, M.A., Ali, A.H. Intrusion detection systems, issues, challenges, and needs. Int. J. Comput. Intell. Syst., 2021.

4. Alzaqebah, A., Aljarah, I., Al-Kadi, O., Damaševičius, R. Modified Grey Wolf Optimization Algorithm for IDS. Mathematics, 2022.

5. Ambusaidi, M.A., He, X., Nanda, P., Tan, Z. Building an IDS using a filter-based feature selection algorithm. IEEE Trans. Comput., 2016.

6. Canadian Institute For Cybersecurity. CICFlowMeter-V4.0 for anomaly detection. Link

7. Chimphlee, S., Chimphlee, W. Machine learning to improve anomaly-based network intrusion detection. Indones. J. Electr. Eng. Comput. Sci., 2023.

8. Gautam, R.K.S., Doegar, E.A. An ensemble approach for IDS using machine learning algorithms. 2018 8th International Conference on Cloud Computing.

9. IDS 2018 Datasets, Canadian Institute for Cybersecurity. Link

10. Jaradat, A.S., Barhoush, M.M., Easa, R.B. Machine learning approach for network intrusion detection. Indones. J. Electr. Eng. Comput. Sci., 2022.

11. Kaja, N., Shaout, A., Ma, D. Intelligent intrusion detection system. Appl. Intell., 2019.

12. Karatas, G., Demir, O., Sahingoz, O.K. Improving IDS performance on an imbalanced dataset. IEEE Access, 2020.

13. Khan, M.A. HCRNNIDS: Hybrid convolutional recurrent neural network-based IDS. Processes, 2021.

14. Kim, J., Shin, Y., Choi, E. IDS based on a Convolutional Neural Network. J. Multimed. Inf. Syst., 2019.

15. Malliga, S., Nandhini, P.S., Kogilavani, S.V. Review of deep learning techniques for DoS attack detection. Inf. Technol. Control, 2022.

16. Momand, A., Jan, S.U., Ramzan, N. Survey of IDS using machine learning, deep learning, datasets, and attack taxonomy. J. Sens., 2023.

17. Muhsen, A.R., Jumaa, G.G., Bakri, N.F.A., Sadiq, A.T. Feature selection strategy for NIDS using Meerkat Clan Algorithm. Int. J. Interact. Mob. Technol., 2021.

18. Nassif, A.B., Talib, M.A., Nasir, Q., Dakalbab, F.M. Machine Learning for Anomaly Detection: A Systematic Review. IEEE Access, 2021.

19. patator | Kali Linux Tools. Link

20. Qusyairi, R., Saeful, F., Kalamullah, R. Ensemble learning and feature selection for improved IDS performance. IAICT, 2020.

21. Songma, S., Sathuphan, T., & Pamutha, T. Optimizing IDS on the CSE-CIC-IDS-2018 dataset. Computers, 2023. DOI