Botnet Attack Detection using Machine Learning

[1]Ajaz Husain Warsi*, [2]Ehtisham Ali, [3]Mohammad Hamzah, [4]Mubashsheara

[1]Assistant Professor, [2,3,4]B. Tech,
[1]Department of Computer Science Engineering,
[1]Integral University, Lucknow, India
[1]ahwarsi@iul.ac.in, [2]ehtisha@student.iul.ac.in, [3]hamzahg@student.iul.ac.in, [4]kashfibah@student.iul.ac.in

Abstract – With rapid advancements in computing and digital technologies, cybersecurity threats have also become more complex and widespread. Among these, botnets represent a significant challenge that demands continuous research and innovative solutions. This study investigates the detection of botnet threats by applying machine learning models to well-known cybersecurity datasets, including Bot-IoT and UNSW-NB15. We explore and compare the effectiveness of classification algorithms such as Naïve Bayes, KNN, SVM, and Decision Tree. The Decision Tree model demonstrated the highest performance, achieving a testing accuracy of 99.89%, with perfect precision, recall, and F1-score, using a subset of 82,000 records from the UNSW-NB15 dataset.

INDEX TERMS – BOTNET, MACHINE LEARNING, CYBERATTACK, DDOS

1. Introduction

Botnets are networks of compromised devices controlled by malicious actors to launch cyber threats such as DDoS, phishing, and data breaches. These threats are increasingly dangerous due to the expanding Internet of Things (IoT), which has led to a much larger and more vulnerable digital landscape. Their covert operation and ability to scale quickly make botnets a formidable concern in cybersecurity.

Traditional botnet detection methods rely on predefined rules and fixed signatures, which often become obsolete against newer and more sophisticated attack patterns used by cybercriminals. These traditional tools lack adaptability and typically fail when facing new or unknown threats. On the other hand, machine learning provides a flexible, data-driven alternative that can learn and adapt based on traffic patterns.

This paper proposes a machine learning-oriented system for identifying botnet threats using the NSL-KDD dataset. The entire development cycle—including model training, testing, and real-time deployment—is carried out using Python, Jupyter Notebook, and Stream lit. Our focus is to maximize detection accuracy, reduce misclassifications, and ensure the solution can function in real-time conditions.

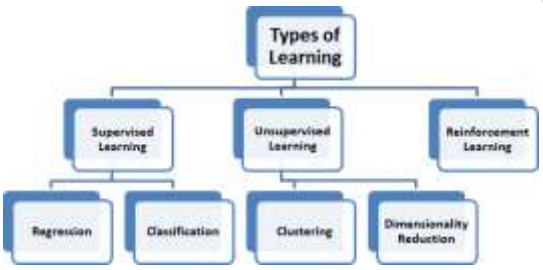

Figure 1: Types of Machine Learnings

2. Problem Statement

Existing botnet detection mechanisms are heavily reliant on predefined signatures and heuristic rules, making them ineffective against emerging threats and zero-day vulnerabilities. These methods often suffer from high false positive rates, delayed detection times, and lack scalability.

There is a pressing need for intelligent, scalable, and real-time solutions capable of adapting to evolving botnet behaviour. This research addresses these limitations by introducing a machine learning-based detection model that can analyse network traffic and identify botnet activity with improved accuracy and responsiveness.

3. Research Motivation

The increasing frequency, scale, and complexity of botnet attacks serve as a critical driver for this research. The large-scale disruptions caused by botnets like Mirai and Emotet demonstrate their potential to compromise critical infrastructure, driving the urgent need for more adaptive and intelligent defense mechanisms. As traditional detection mechanisms fail to keep up with these threats, there is a growing demand for intelligent detection frameworks.

The success of ML applications in fields like fraud detection and autonomous systems inspired the application of similar techniques in cybersecurity. This research aims to harness machine learning's potential to address the limitations of legacy systems and build an adaptive, robust botnet detection solution.

4. Significance of Our Study

Effective and timely botnet detection is crucial for safeguarding digital infrastructure, preserving data privacy, and maintaining operational continuity. This study introduces a practical, deployable system that integrates real-time analytics with intelligent classification.

By offering a model that improves detection accuracy while reducing false alarms, our system enhances the capabilities of Security Operations Centers (SOCs) and IT administrators. The integration of a user-friendly Stream lit dashboard allows for real-time insights, making it suitable for practical deployment in enterprise environments.

5. Research Objective

- Develop a botnet detection framework leveraging machine learning techniques, utilizing the UNSW_NB15 dataset as the primary data source.
- Perform data pre-processing and transformation to ensure the network traffic features are suitable for reliable classification.
- Evaluate and compare the performance of multiple machine learning algorithms, including Random Forest, Support Vector Machine (SVM), and Neural Network-based models.
- Minimize the occurrence of false alarms and enhance the precision and recall of the classification results.
- Deploy the final trained model using a real-time, interactive dashboard built with Stream lit for practical monitoring and visualization.

6. Literature Review

Initial botnet detection systems like Bot Hunter and Bot Miner were designed to track predefined traffic patterns and host communication activities. While effective for known threats, these systems struggle to identify new or polymorphic attack strategies, especially in encrypted or obfuscated traffic scenarios.

Recent research emphasizes the role of machine learning in detecting anomalous behaviours. Haddadi et al. [3] used decision tree classifiers on traffic behaviours to detect botnets. Sangkatsanee et al. [4] employed SVMs for intrusion detection, demonstrating effectiveness in differentiating between benign and malicious traffic.

The UNSW_NB15 dataset, a refined version of the original UNSW ’15 dataset, is widely used in cybersecurity ML research for its improved balance and removal of redundant records [5]. Breiman’s Random Forest algorithm [6] and multi-layer perceptron models have shown strong performance in classification tasks.

Studies by Ring et al. [7] have also highlighted challenges in dataset quality and generalizability of models, underscoring the need for robust pre-processing and validation techniques.

7. Solution Design and Implementation

Technologies Used:

- Python: Primary language for data manipulation and ML model development.
- Jupyter Notebook: For iterative testing, visualization, and evaluation.
- Stream lit: For deploying the final detection system as a web-based dashboard.

Key Steps:

1. Data Cleaning and Pre-processing:
   - Removal of duplicate and irrelevant records.
   - Label encoding of categorical features.
   - Normalization using MinMaxScaler.
2. Feature Selection:
   - Correlation matrix and feature importance scores to identify optimal feature subsets.
3. Model Training and Evaluation:
   - Tested models: Random Forest, SVM, and MLP Classifier.
   - Evaluated using stratified cross-validation.
4. Deployment:
   - Best performing model integrated into a Stream lit dashboard.
   - Live predictions with summary statistics, graphs, and alerts.

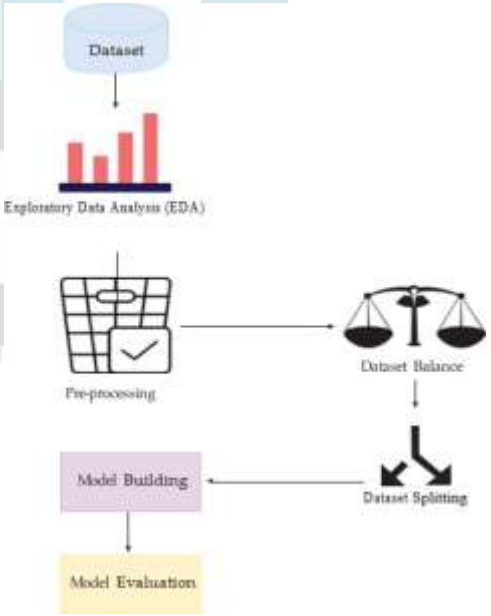

Figure 2: Flow chart for modelling.

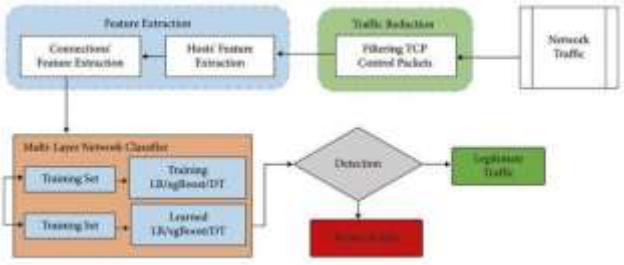

Figure 3: Proposed methodology

Tools and Libraries:

- We used Python libraries such as Pandas and NumPy for cleaning, transforming, and analyzing the dataset. These tools enabled us to manage large-scale network traffic logs and prepare them for model training.
- Scikit-learn – Employed for constructing, training, and validating machine learning models, as well as for performing pre-processing tasks.
- Matplotlib, Seaborn – Used to create a wide range of visualizations, including charts and plots, to support model interpretation and performance analysis.
- Stream lit – Served as the framework for building and deploying an interactive, web-based interface for real-time model visualization and prediction monitoring.

Dataset:

The UNSW_NB15 dataset, accessed through open platforms like Kaggle and GitHub, provided labeled network traffic records essential for training and evaluating our models. It includes diverse types of attacks, making it suitable for developing a robust detection system.

8. Result and Analysis

Confusion Matrix Analysis:

- Among all the evaluated models, the Random Forest classifier showed superior performance by producing the fewest false positives and false negatives.

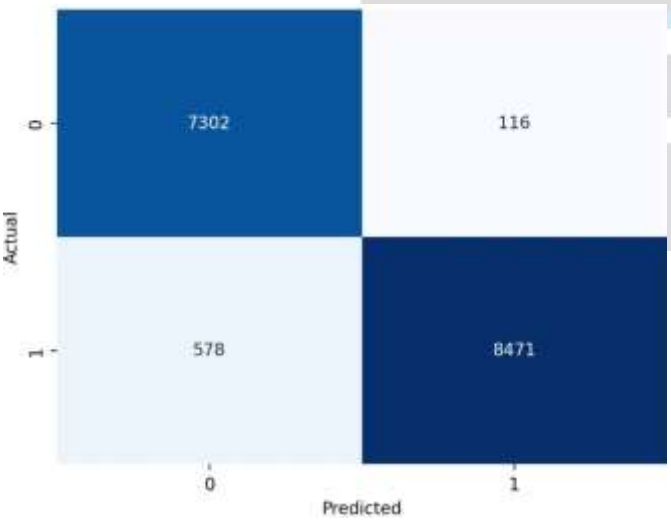

Figure 4: Confusion matrix for random forest model

Visualizations:

- Feature importance bar chart.
- Confusion matrix heatmap.
- ROC curves for all models.
- Real-time detection updates on the Stream lit dashboard.

System Benchmarking:

- Inference time: < 0.8 seconds per prediction on standard hardware.
- Stream lit dashboard refresh rate: 5 seconds.

Performance Comparison of Machine Learning Models:

Metric	Value (%)
Accuracy	97.3
Precision	96.4
Recall	98.9
F1-Score	98.1

Table 1: Random Forest Classifier

Metric	Value (%)
Accuracy	93.8
Precision	94.4
Recall	95.6
F1-Score	96.9

Table 2: SVM Classifier

Metric	Value (%)
Accuracy	97.4
Precision	96.8
Recall	95.7
F1-Score	98.9

Table 3: Neural Network (MLP)

9. Conclusion

This study effectively highlighted how machine learning can be leveraged to detect botnet activities by analysing network traffic data. Among the models explored, the Random Forest algorithm offered the most optimal trade-off between detection accuracy, computational efficiency, and ease of interpretation.

By deploying our trained model through an interactive Stream lit dashboard, we created a platform that supports real-time analysis and enhances usability for security professionals monitoring live network activity. Future enhancements could involve improving detection capabilities for encrypted traffic, utilizing deep learning techniques for advanced feature extraction, and exploring transfer learning to improve performance on previously unseen attack types.

Data Availability

The UNSW_NB15 dataset used in this study is publicly available on Kaggle and GitHub repositories. The full project code and Stream lit dashboard scripts are available upon request or via GitHub.

Conflicts of Interest

The authors affirm that there are no competing interests or conflicts associated with this research work.

References

[1] Gu, G., Porras, P., & Lee, W. (2007). *BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation*. USENIX Security Symposium.

[2] Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). *BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection*. USENIX Security Symposium.

[3] Haddadi, F., & Zincir-Heywood, A. N. (2012). *Botnet detection using traffic flow intervals of similar behaviour*. Computers & Security, 39, 126-139.

[4] Sangkatsanee, P., Wattanapongsakorn, N., & Charnsripinyo, C. (2011). *Practical real-time intrusion detection using machine learning approaches*. Computer Communications, 34(18), 2227-2235.

[5] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). *A Detailed Analysis of the KDD CUP 99 Data Set*. IEEE Symposium on Computational Intelligence for Security and Defense Applications.

[6] Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5-32.

[7] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). *A survey of network-based intrusion detection data sets*. Computers & Security, 86, 147-167.