Application of XGBoost Classification on CTU-13 for Botnet Detection

Adam Sam
School of Computing and Data Science
Wentworth Institute of Technology
550 Huntington Ave, Boston, MA 02115
sama1@wit.edu

***Abstract*—This study investigates the use of Extreme Gradient Boosting (XGBoost) for detecting Distributed Denial-of-Service (DDoS) Botnet traffic by analyzing packet capture files, building on prior applications. Using data derived from the CTU-13 dataset—specifically Scenario 4, which captures DDoS attacks—we construct a feature-rich dataset by merging packet-level .pcap data and bidirectional NetFlow summaries. The inclusion of fine-grained features enables the model to capture nuanced traffic behavior belonging to three categories: Botnet, Legitimate, and Background. Our XGBoost classifier achieves a test accuracy of 99.82%, with high F1-scores for Background (99.90%) and Legitimate (97.70%) traffic. Botnet detection performance, however, was found to be weaker (F1-score: 89.65%). This reveals challenges tied to limited botnet representation in the dataset. Feature importance analysis reveals the relevance of TCP flags—particularly the ACK flag—in identifying SYN-flood related botnet behavior. Despite training on only a subset of the entire dataset, these results validate the applicability of tree-based learning methods to traditional network intrusion detection systems (NIDS).**

***Index Terms*—CTU-13, packet capture, logs, machine learning, predictive modeling, classification, XGBoost**

I. Introduction

The widespread adaptation of Artificial Intelligence (AI) and Machine Learning (ML) has transformed numerous domains, including cybersecurity [1]. These technologies have enabled new approaches to threat detection by leveraging complex datasets to uncover malicious activity through pattern recognition and predictive modeling. In particular, ML has proven to be effective in the realm of network intrusion detection systems (NIDS), where traditional signature-based methods often underperform in the identification of novel or evolving threats [2].

Botnets (networks of compromised devices controlled by malicious actors) in particular pose significant security risks, including data breaches, denial-of-service attacks, and spam. Existing NIDS, especially those deployed in real-time environments, face challenges related to scalability, latency, and detection rates [3].

In recent times, machine learning models have emerged as promising tools in NIDS, allowing for the identification of suspicious traffic based on intricate patterns and real-time behaviors [4]. Among these, tree-based ensemble methods such as XGBoost have shown remarkable promise due to their ability to handle large-scale data, model complex relationships, and perform efficiently under constrained computational environments [5] [6]. Compared to other classification techniques such as support vector machines (SVMs), random-forest, and K-means clustering, XGBoost in particular has consistently demonstrated superior performance in terms of predictive accuracy [7]. Previous work has already demonstrated the effectiveness of XGBoost in detecting botnet activity in Internet of Things (IoT) environments [8]. However, outside of the domain of IoT, the applicability of XGBoost to a more conventional network environment comprising traditional computing systems remains unexplored.

This study aims to evaluate the performance of XGBoost in detecting botnet traffic using the CTU-13 dataset, which simulates realistic attack scenarios in conventional computer networks [9]. By training and validating the model on labeled PCAP (packet capture) logs, we assess whether XGBoost can maintain high classification accuracy outside the context of IoT environments. The study replicates and extends methodologies from recent literature, with a particular emphasis on comparative analysis against IoT-focused models.

The remainder of this paper is organized as follows: the Related Works section outlines the problem definition and distinguishes it from related work. The Results section briefly details the methodology and analyzes the results. Finally, the Conclusion section discusses conclusions and future work.

II. Related Works

To address the limitations with existing NIDS, recent research has turned towards ensemble learning techniques, most notably the Extreme Gradient Boosting (XGBoost) technique, which has shown promising results in cybersecurity-related classification tasks due to its computational efficiency and ability to accurately model complex datasets [10].

H. Ravikumar [8] proposed an XGBoost-based botnet detection architecture specifically tailored for IoT networks operating within cloud platforms. Their results show that the XGBoost classifier achieved high accuracy (9.88 F1-score) in detecting botnet behaviors in IoT traffic. Ravikumar's work lays the foundation for this study, which builds upon their methodology by applying the same techniques on a different environment: conventional network settings, using packet capture data from CTU-13.

W. Niu et al. [10] demonstrated the effectiveness of XGBoost in detecting infected hosts from HTTP traffic by similarly utilizing packet capture data from the UNSW-NB15 dataset. Their system achieved high throughput (1859 detections per second), a 98.72% accuracy rate, and a false positive rate below 1%. While the UNSW-NB15 dataset is similar to the CTU-13 in which it contains packet capture data, CTU-13 is distinguished by its organization of its data into thirteen different botnet scenarios, including "spam" propagation and distributed denial-of-service (DDoS) attacks. This diversity allows for a more comprehensive evaluation of detection models across different types of malicious behavior as well as the possibility of a multi-class approach for classification. Our study uses this advantage to test whether XGBoost can generalize well across various threat types. We expect our approach to achieve comparable performance when trained using the CTU-13 dataset.

R. Terado and M. Hayashida [2] explored the comparative performance of XGBoost and Random Forest in NIDS using the Kyoto 2016 dataset. Their findings demonstrate that XGBoost not only delivers higher prediction accuracy over Random Forest, but also significantly reduces inference time, which is critical for real-time threat mitigation. This paper is instrumental in affirming the choice of XGBoost over other ensemble methods of anomaly classification.

These prior works collectively highlight the strengths of XGBoost in cybersecurity applications. However, they leave open the question of how well such models generalize in more broader contexts. This paper seeks to fill that gap solely through our choice of using the CTU-13 dataset.

III. Methods

A. Dataset

The CTU-13 dataset was collected by the CTU University in the Czech Republic in 2011 and is composed of thirteen distinct network capture scenarios, each involving a specific malware sample executing different behaviors across various protocols [9]. The intent of the dataset is to capture a realistic mix of botnet, legitimate, and background traffic. Each scenario includes full packet captures (.pcap) and corresponding bidirectional flow-level summaries (.netflow) containing the labels for each packet.

Due to hardware limitations, we processed only a subset of the available data. This study focuses solely on Scenario 4, which captures Distributed Denial of Service (DDoS) activity. The entire CTU-13 dataset totaled hundreds of gigabytes. The full converted .pcap file for only the Scenario 4 dataset alone yielded approximately 62 million packets and totaled over 21 GB. To enable processing within limited system memory, we restricted conversion and analysis to the first 6.2 million packets, resulting in a final working dataset of approximately 1.63 GB, or roughly 10% of the total available capture. While this reduction limits the scope of the analysis, the retained portion still offers a sufficient sample of traffic patterns for meaningful evaluation.

To build a feature-rich dataset, we combine the packet capture file with the NetFlow file. Packet-level data was extracted in CSV form from the dataset's .pcap file using TShark. This data includes TCP flags, window sizes, and other fine-grained network attributes, which serve as the features we train our model on. The .netflow file contains bidirectional flow-level summaries containing the labeled traffic classes which our model will predict. This file was converted to CSV and combined using Python's Pandas library [11].

Both datasets were combined using a fuzzy-based merge [12] using timestamps, and source and destination addresses as a unique identifier. A challenge in this process was that the timestamps in the two datasets did not perfectly line up. Since both datasets reported time in different formats (.pcap provided epoch time while NetFlow provided a date-time string), we had to run an additional epoch-timestamp conversion on the NetFlow's date-time data - causing slight timestamp discrepancies between the two datasets. To solve this, the merge was performed by using Pandas' pd.merge_asof function, designed to combine two datasets based on their nearest values, effectively allowing a margin of error for the timestamp values.

Table I provides a description of all the features within the dataset. The final labeled dataset includes three classes:

- Botnet: Traffic originating from infected hosts.
- LEGITIMATE: Clean user-generated network activity.
- Background: Ambient or unidentified traffic unrelated to botnet activity.

Data was split into 70% training and 30% testing. Initial analysis of class balance, presented in Table II showed a significant imbalance, with over 5.8 million Background, 232,800 Legitimate, and only 44 Botnet samples. To address this imbalance, we applied SMOTE (Synthetic Minority Oversampling Technique) [13] on our training data, equalizing the set to roughly 4.09 million instances per class. All features were then normalized and encoded appropriately, producing balanced dataset with 21 features for training and evaluation.

B. Model Training

Python's scikit-learn library was used for model training, data splitting, as well as cross-validation using 3-fold Bayesian optimization [14]. Considering the large size of our dataset, this method was chosen over a full grid search due to its greater time-efficiency in exploring the hyperparameter space [15]. The search space was configured to search for the optimal values for several hyperparameters, including the number of estimators, the maximum tree depth, the learning rate, and various regularization parameters. Upon completion of the search, the best-performing set of hyperparameters was used to fit the final XGBoost model onto the training set.

TABLE I
MERGED DATASET FEATURES

Feature	Description
ip.proto	IP protocol number (e.g., 6 for TCP, 17 for UDP, 1 for ICMP).
frame.len	Total length of the captured frame in bytes.
tcp.flags	TCP flags as a hexadecimal value.
tcp.flags.syn	Boolean flag indicating if the SYN (synchronize) flag is set.
tcp.flags.ack	Boolean flag indicating if the ACK (acknowledgment) flag is set.
tcp.flags.reset	Boolean flag indicating if the RST (reset) flag is set.
tcp.flags.push	Boolean flag indicating if the PSH (push) flag is set.
tcp.flags.fin	Boolean flag indicating if the FIN (finish) flag is set.
tcp.window_size_value	Size of the TCP receive window.
icmp.type	Type of ICMP message.
icmp.code	Code of the ICMP message.
ip.ttl	Time To Live of the IP packet.
source_port	Source port number of the communication.
destination_port	Destination port number of the communication.
duration	Duration of the network flow in seconds.
flag	TCP flags observed in the network flow.
tos	Type of Service field from the IP header.
packets	Number of packets in the network flow.
bytes	Number of bytes in the network flow.
flows	Number of flows.
label	Class label for the traffic (Background, LEGITIMATE, or Botnet).

TABLE II
CLASS COUNTS BEFORE AND AFTER SMOTE

Class	Count Pre-SMOTE	Count Post-SMOTE
Background	5,842,176	4,089,523
LEGITIMATE	232,800	4,089,523
Botnet	44	4,089,523

IV. RESULTS

A. Model Performance

Our preliminary experimentation with XGBoost-based classification of botnet traffic yielded promising results, but also highlighted several important caveats that arise from using a reduced dataset. After training, our XGBoost model achieved an overall accuracy of 99.82% on the test set. The F1-scores for the majority classes, “Background” and “LEGITIMATE”, were 99.90% and 97.7%, respectively. The model’s performance on the Botnet class was weaker, however, with an F1-score of 89.65%. The weak botnet performance most likely stems from its under-representation in the dataset, despite the oversampling performed by SMOTE. The classification report is shown in Table III.

TABLE III
CLASSIFICATION REPORT

Class	Precision	Recall	F1-Score
Background	0.9999	0.9983	0.9991
Botnet	0.8125	1.0000	0.8966
LEGITIMATE	0.9593	0.9964	0.9775
Macro Avg	0.9239	0.9982	0.9577
Weighted Avg	0.9983	0.9982	0.9983

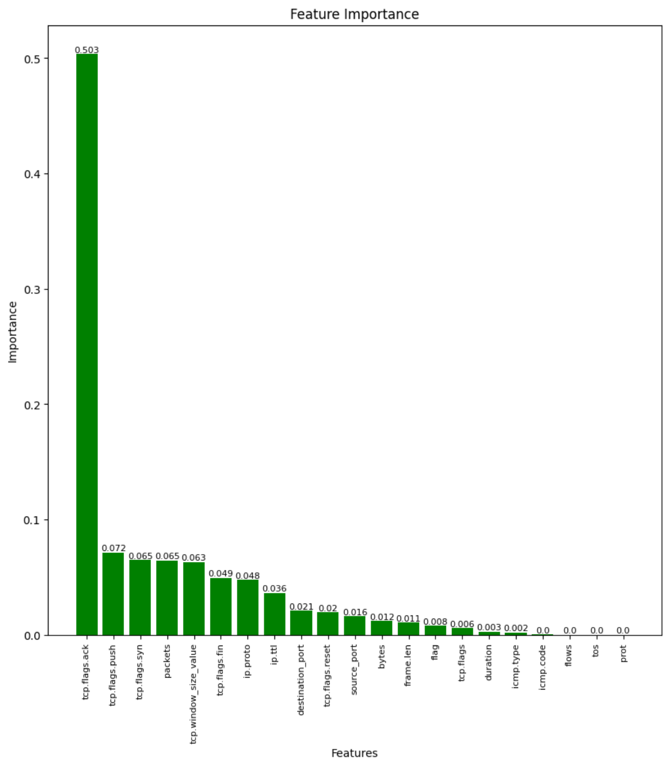

Fig. 1. Feature importance graph. The TCP ”ACK” flag has the highest influence over model predictions.

B. Interpretation

Figure 1 shows feature importance obtained using XGBoost’s built-in method. The most influential feature by a significant margin is ”tcp.flags.ack”. This finding aligns with what is commonly seen in DDoS attacks - SYN floods.

In SYN flood attacks, a botnet will send a large amount of SYN packets to the target. The target responds with SYN-ACK packets and waits for the final ACK to complete the three-way handshake [16]. However, the bots in the botnet never send the final ACK. The model effectively learns that a high volume of traffic with the SYN flag but without a corresponding ACK is a strong indicator of a SYN flood attack. The presence or absence of the ACK flag becomes a critical feature for distinguishing legitimate traffic from this type of attack. Other notable features include tcp.flags.push and tcp.flags.syn, further emphasizing the significance of TCP flag analysis in identifying botnet traffic.

V. Conclusion

This study demonstrates the viability of using XGBoost for botnet traffic classification using the CTU-13's DDoS scenario dataset. By leveraging a dataset that combines packet-level and flow-level features, the model was able to achieve strong overall performance, attaining a test accuracy of 99.82%, a macro average F1-score of 95.77%, and a weighted average F1-score of 99.83%. F1-scores for the Background and LEGITIMATE classes were 99.90% and 97.70%, respectively, indicating that the model is highly effective at distinguishing normal and ambient traffic.

However, the model's performance on the Botnet class was comparatively weaker, with an F1-score of 89.65%. This limitation is largely attributed to the large scarcity of Botnet samples in dataset. This is further exasperated by our choice of using a small subset of the entire dataset. While oversampling with SMOTE was performed to mitigate this issue, the underlying class imbalance and constrained training data still limited the model's ability to generalize effectively on botnet traffic patterns.

Feature importance analysis revealed that TCP flags, especially tcp.flags.ack, were critical in identifying DDoS-related botnet activity. This aligns with known characteristics of SYN-flood attacks, in which botnets initiate incomplete TCP handshakes by withholding the final ACK. The model effectively learned to associate patterns in TCP flag usage with malicious behavior, confirming the utility of protocol-level feature analysis in detection tasks.

While these initial results show promise, the current study is constrained by hardware limitations and dataset scope. Future work would focus on scaling the analysis to a larger subset—or the entirety—of the CTU-13 dataset, encompassing all thirteen scenarios and a broader spectrum of botnet behaviors beyond DDoS attacks. Doing so would provide the model with more training examples, particularly of Botnet-labeled traffic, thereby improving predictive power.

References

[1] J. Martínez Torres, C. Iglesias Comesana, and P. J. García-Nieto, "Machine learning techniques applied to cybersecurity," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2823–2836, 2019.

[2] R. Terado and M. Hayashida, "Improving accuracy and speed of network-based intrusion detection using gradient boosting trees," in *International Conference on Information Systems Security and Privacy*, 2020.

[3] C. Zhang, X. Costa-Pérez, and P. Patras, "Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms," *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1294–1311, 2022.

[4] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[6] D. Nielsen, "Tree boosting with xgboost-why does xgboost win" every" machine learning competition?," Master's thesis, NTNU, 2016.

[7] K. M. K. Raghunath, V. V. Kumar, M. Venkatesan, K. K. Singh, T. R. Mahesh, and A. Singh, "Xgboost regression classifier (xrc) model for cyber attack detection and classification using inception v4," *Journal of Web Engineering*, vol. 21, no. 4, pp. 1295–1322, 2022.

[8] H. Ravikumar, "International journal of multidisciplinary and current research xgboost-based botnet detection architecture for iot networks in cloud platforms," *International Journal of Multidisciplinary and Current Research*, vol. 12, pp. 453–460, 07 2024.

[9] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers Security*, vol. 45, pp. 100–123, 09 2014.

[10] W. Niu, T. Li, X. Zhang, T. Hu, T. Jiang, and H. Wu, "Using xgboost to discover infected hosts based on http traffic," *Security and Communication Networks*, vol. 2019, no. 1, p. 2182615, 2019.

[11] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[12] M. Cayrol, H. Farreny, and H. Prade, "Fuzzy pattern matching," *Kybernetes*, vol. 11, no. 2, pp. 103–116, 1982.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[15] I. Dewancker, M. McCourt, and S. Clark, "Bayesian optimization for machine learning: A practical guidebook," *arXiv preprint arXiv:1612.04858*, 2016.

[16] M. Bogdanoski, T. Suminoski, and A. Risteski, "Analysis of the syn flood dos attack," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 5, no. 8, pp. 1–11, 2013.