AI-Powered Malware Detection using Network Flow Analysis

Osama Azab
Prince Sultan University
Riyadh, Saudi Arabia
221110390@psu.edu.sa

ABSTRACT

This study presents a machine learning-based approach to detect and classify malicious network traffic using NetFlow-derived features. We utilize real-world traffic from the CTU-13 botnet dataset and CTU-Normal benign dataset. A cleaned and merged dataset of over 16 million flow records was engineered and used to train multiple models for binary and multi-class classification. We evaluate Logistic Regression, XGBoost, LSTM, and Linear SVM. Results show that XGBoost and LSTM substantially outperform linear baselines, achieving macro F1 scores of 0.9648 and 0.8923, respectively. Our work highlights the feasibility of deploying intelligent, flow-based intrusion detection systems in real-time environments.

ACM Reference Format:
Osama Azab. 2025. AI-Powered Malware Detection using Network Flow Analysis. In . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Cyberattacks such as botnets, DDoS assaults, and data exfiltration campaigns pose increasing threats to organizations worldwide. Traditional intrusion detection systems (IDS), which rely on predefined rules or static signatures, often fail to detect new or obfuscated attacks. Consequently, there is a growing demand for adaptive, data-driven solutions.

Machine learning (ML) techniques offer a compelling alternative. By learning from patterns in historical traffic data, ML models can detect anomalies, identify malware behavior, and classify different types of malicious activity. However, effective deployment of ML-based IDS systems requires large-scale datasets, careful feature engineering, and thorough validation.

This research focuses on detecting and classifying cyberattacks based on NetFlow-like features using ML models. We employ datasets from the Stratosphere IPS project—namely, CTU-13 (malware-infected traffic) and CTU-Normal (benign traffic)—to build a comprehensive flow-based dataset. We evaluate multiple classification algorithms on both binary (malicious vs. benign) and multi-class (malware type) tasks. Our results reveal that gradient boosting and deep learning models significantly outperform traditional baselines, particularly in handling class imbalance and recognizing minority threat categories.

Conference'17, July 2017, Washington, DC, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 DATASET AND PREPROCESSING

The dataset used in this work was created by combining two publicly available resources: the CTU-13 botnet traffic dataset and the CTU-Normal benign traffic dataset, both maintained by the Stratosphere IPS project. These datasets include labeled NetFlow-style records representing communication flows between endpoints.

Each dataset was downloaded, extracted, and converted into CSV format. All available .binetflow files were merged into a unified DataFrame. Timestamps were parsed and standardized using datetime encoding. Network ports, protocols, and state fields were cleaned and made consistent across both datasets.

Missing or malformed entries (e.g., empty ports or timestamps) were either corrected or dropped. Port numbers were converted to integers, and invalid ports were replaced with -1. The flow state column, which often contained missing or inconsistent values, was filled with a placeholder "UNKNOWN" where necessary.

To prepare the target variables, two versions of the label were created. First, a binary label distinguishing between benign and botnet traffic was assigned. Second, a multi-class label was constructed by parsing the original label strings to identify malware types, such as Botnet, DNS, NTP, IRC, and others. This was done using regular expression matching and domain-specific keyword grouping.

The result was a well-structured dataset containing over 16 million labeled network flows, each with temporal, volumetric, and structural features ready for feature engineering and modeling.

3 FEATURE ENGINEERING AND SELECTION

The final dataset included a wide range of features extracted from NetFlow records. These features describe various aspects of a flow, including its duration, volume, directionality, and structure. Feature engineering focused on enhancing the discriminative power of the dataset while minimizing redundancy and overfitting risk.

We retained key raw features such as Duration, TotPkts, TotBytes, SrcBytes, and type-of-service fields (sTos, dTos). Next, we constructed new features using domain knowledge, including:

- PktByteRatio = Total packets / Total bytes
- BytePerPkt = Total bytes / Total packets
- SrcByteRatio = Source bytes / Total bytes

Low variance filtering was applied to eliminate features with little variability. Pearson correlation analysis was then used to remove highly correlated features (correlation coefficient > 0.95). In such cases, only one representative feature from each pair was retained.

The final feature set included eleven numerical fields that capture packet volume, byte ratios, port values, and structural characteristics. These were used as input vectors for all models.

4 DATA EXPLORATION AND CORRELATION ANALYSIS

Prior to modeling, exploratory data analysis (EDA) was conducted to better understand feature behavior. Summary statistics indicated highly skewed distributions in duration and packet count. Most flows were short, low-volume connections, but some long and dense flows also existed, resulting in high variance.

To evaluate redundancy, we calculated a correlation matrix across numeric features. Key correlations:

- TotPkts strongly correlates with TotBytes and SrcBytes
- BytePerPkt inversely correlates with TotPkts
- SrcByteRatio and PktByteRatio had lower correlations, offering unique signal

A heatmap was generated to visualize these correlations and served as a guide for pruning.

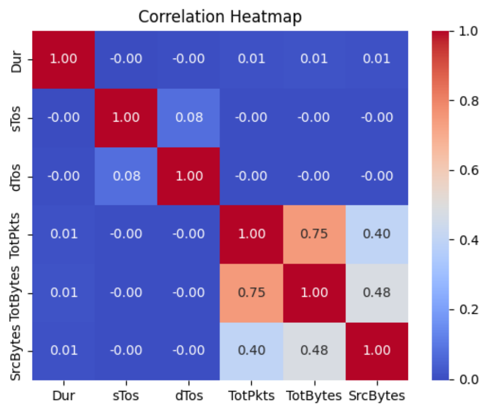

Figure 1: Correlation matrix heatmap of engineered features.

5 LABEL DISTRIBUTION AND CLASS IMBALANCE

The binary classification task was framed as:

- Label 0: Benign (includes Background and Normal)
- Label 1: Botnet

Counts:

- Benign: 19.76 million samples
- Botnet: 444.7 thousand samples

For the multi-class task, traffic was grouped into categories. Counts included:

- Background: 19,175,582
- Normal: 586,473
- Botnet: 444,699
- NTP: 1,268

The dataset was extremely imbalanced, with over 95% of samples belonging to the benign Background and Normal categories. This imbalance led us to favor macro-averaged performance metrics and to explore models resilient to data skew.

6 MODELING APPROACH

We implemented and evaluated four machine learning models for both binary and multi-class classification:

- Logistic Regression (LR)—Baseline linear model with probabilistic output.
- Linear Support Vector Machine (SVM)—Margin-based linear classifier.
- XGBoost—Ensemble of gradient-boosted decision trees.
- LSTM Neural Network—Sequential model capturing temporal dependencies using PyTorch.

Each model was trained using an 80/10/10 split (train/val/test), stratified by label. LSTM input was shaped appropriately (1 timestep) with float32 tensors. Early stopping was used in LSTM training to avoid overfitting.

Performance metrics included:

- Accuracy
- Weighted Precision, Recall, and F1-Score
- Macro-averaged Precision, Recall, and F1-Score

7 RESULTS

7.1 Binary Classification

Model	Accuracy	Weighted F1	Macro F1
Logistic Regression	97.80%	0.9672	0.4945
Linear SVM	97.80%	0.9672	0.4945
XGBoost	98.06%	0.9807	0.7766
LSTM	99.07%	0.9903	0.8824

Table 1: Binary classification performance comparison.

7.2 Multi-Class Classification

Model	Accuracy	Weighted F1	Macro F1
Logistic Regression	97.80%	0.9672	0.4945
Linear SVM	97.80%	0.9672	0.4945
XGBoost	99.70%	0.9970	0.9648
LSTM	99.01%	0.9904	0.8923

Table 2: Multi-class classification performance comparison.

8 DISCUSSION

Our results highlight the importance of model selection when dealing with imbalanced and multi-class network traffic data. Simple linear models such as Logistic Regression and SVM, while fast and interpretable, fail to generalize to underrepresented malware classes. Their macro F1-scores remain below 0.5 despite high overall accuracy, indicating overfitting to the dominant Background and Normal traffic classes.

XGBoost consistently achieved top performance in both binary and multi-class scenarios. Its ability to construct nonlinear decision boundaries and prioritize difficult examples using boosting mechanisms made it ideal for detecting subtle attack flows. It also scaled efficiently to millions of records, making it a practical candidate for production use.

LSTM, while slightly more resource-intensive to train, also performed remarkably well. Its temporal modeling capabilities gave it an advantage in detecting complex behavior patterns across flow features. LSTM achieved the highest macro recall and strong performance on minority classes.

9 CONCLUSION AND FUTURE WORK

This project demonstrates the viability of machine learning-based malware detection using flow-level features. By leveraging CTU-13 and CTU-Normal datasets, we built a large and well-structured dataset suitable for supervised classification. Our evaluation of multiple models showed that while traditional models offer quick baselines, XGBoost and LSTM significantly outperform them on critical metrics like macro F1-score and minority class detection.

Future directions include:

- Integrating temporal/session-based aggregation for richer modeling.
- Applying data resampling or cost-sensitive learning to further improve minority class detection.
- Exploring ensemble models that combine the strengths of tree-based and deep learning techniques.
- Incorporating real-time stream processing frameworks (e.g., Apache Kafka, Spark Streaming).

10 CONTRIBUTIONS

This project was conducted entirely by Osama Gamal Azab. The following contributions were made:

- Collected and preprocessed the CTU datasets.
- Engineered and selected features.
- Implemented and trained all machine learning models.
- Evaluated model performance and interpreted results.
- Built visualizations and wrote this report.

11 ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Almuthanna Alageel for his guidance, feedback, and support throughout the course of this project.

REFERENCES

[1] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. 2014. An empirical comparison of botnet detection methods. Computers & Security, 45: 100–123.
[2] Stratosphere IPS Project. CTU-13 and CTU-Normal Datasets. https://www.stratosphereips.org
[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD. 785–794.
[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press.