Malware attack detection using network flows with machine learning

Abdul Razzak R Yergatti[1], Prajwal Shiggavi[1], Mohammed Azharuddin[1], and Suneeta V Boodihal[1]

KLE Technological University Hubballi, India;
School of Electronics and Communication Engineering.
abdulrazzakryergatti@gmail.com, mohammed.azharuddin@kletech.ac.in

Abstract. Traditional defense solutions like intrusion detection and thorough packet inspection are not so accurate. These techniques include signature-based detection, which uses known patterns, and heuristic or behavioral analysis, which evaluates program behavior to detect suspect activities. The demand for more advanced and continuously innovative methods to combat malware, botnets, and other malicious activities is urgent. Machine Learning (ML) emerged as a promising approach due to increasing computing power and reduced costs, offering potential as either an alternative or complementary defense mechanism to enhance detection accuracy by learning from large datasets of known malware behaviors.

blindtext

Keywords: · Malware detection · Network flow · Gradient Desent · Random forest · Dense Neural Network.

1 Introduction

Cybersecurity is developing and the level of online crimes is constantly increasing. Because sophisticated assaults are growing more common and extensive, they are regarded as the new normal. The cybersecurity defense must likewise innovate in response to this ongoing shift. There are already answers, yet these approaches are still often combined. Systems for detecting and preventing network intrusions (IDS/IPS) keep an eye out for hostile activities and policy infractions. Detecting malware that matches recognized signatures is much easier using signature-based systems to detect intrusions. In contrast, an intrusion detection system that is based on behavior analyzes and reports on system abnormalities[1]. Despite their effectiveness, both kinds have some drawbacks. Systems that rely on signatures from recognized threats are useless against zero-day assaults or novel malware samples. Because traditional behavior-based systems rely on a standard profile, which becomes more difficult to establish as networks and applications get more complex, they may not be useful for detecting anomalies. Another approach is full data packet analysis, however, it is risky in terms of sensitive user data exposure and computationally costly[2]. Particularly in the subject of cybersecurity, machine learning (ML) has attracted a lot of interest across a wide range of applications and disciplines. Machine learning techniques may be used to evaluate and categorize undesirable actors from a vast amount of available data, as hardware and processing capacity become

more widely available. Hundreds of machine learning algorithms and techniques may be essentially divided into two categories: supervised and unsupervised learning[3]. Approaches to supervised learning are used in the context of regression, in which input is mapped to a continuous output, or classification, in which input matches to an output. Unsupervised learning has been used in dimension reduction and exploratory analysis, and it is mostly achieved by clustering. By using both of these techniques, cybersecurity can analyze malware almost instantly and do away with the drawbacks of conventional detection techniques.

We analyze NetFlow data with our technique. NetFlow records don't reveal private or personally identifiable information (PII), but they do include sufficient information to identify traffic uniquely using qualities like 5-tuples and other fields[4]. For network administration and monitoring, NetFlow is already widely used in conjunction with its open standard version IPFIX. NetFlow is an effective option because of its privacy characteristics and availability of data.

2 Related literature

The term malware is more specific than intrusion. It covers a range of defined categories such as worms, ransomware, etc. Intrusion, on the other hand, is broader. It also includes an active actor such as an individual behind numerous programs systematically trying to gain access to an asset. When talking about malware, whether anomaly or signature-based detection techniques, there is reliance that some parts of the malware are static for detection. That is, there is a component that is predictable and can be used for detection. The easiest way to understand this is with signature detection. It relies on identifying a part of the program, adding that particular feature to the malware detector, and deploying the solution[5]. Intrusion detection systems can also be malware detectors because they use signature and behavior-based techniques. The difference is in the scope of what is being targeted. With malware, the behavior is more deterministic, with intrusion it is broader and requires further interpretation. In this sense, intrusion detection is more amenable to behavior-based profiles. If some predefined broad rules are triggered, then inspection is merited. That is, an alarm trigger may be malware but also an active intrusion by an actor and it is up to a person or program to try to sort it out. To sum up, a malware program is more specific than an intrusion detection. An easy way to separate them is on the scope of what they are targeting. Be mindful that it may not be easy to draw the distinction with a particular technique or program. It all depends on the particular implementation and what the software is targeting[6].

There are significant drawbacks to using traditional techniques for malware detection. One of the earliest techniques is signature-based detection, in which the system searches for the malware's recognized patterns. This is effective against known threats but cannot detect newly released or modified malware that deviates from established patterns. Heuristic-based detection looks for unusual behavior or code in trying to find malware; this method often advertises benign software as harmful, resulting in many

false alarm rates[7].

A further approach that checks the network for unusual activity is anomaly-based detection. It generates a model of typical network activity and marks any deviation from this model as possibly dangerous. Nevertheless, this strategy may also result in many false positives because it may be challenging to characterize "normal" behavior, particularly in diverse and complicated network contexts. To enhance malware detection in light of these constraints, researchers have turned to machine learning (ML). ML systems can recognize intricate patterns that conventional approaches overlook by learning from vast volumes of data. Additionally, they may adjust to new malware varieties by retraining on fresh data, increasing their effectiveness against ever-changing threats[8].

Supervised and unsupervised learning algorithms are the two primary categories of machine learning algorithms utilized in malware detection. Labeled information is used to train supervised learning systems, with each sample classified as harmful or secure. Standard supervised learning techniques consist of support vector machines (SVM), which determine the best way to separate different classes of data; decision trees, which partition information based on particular traits to make a selection; random forest machines, which combine multiple decision trees to improve accuracy, and neural networks, which are sophisticated models that can recognize highly complex patterns[9]. Conversely, unsupervised learning techniques do not require information to be labeled. They are employed in data analysis to identify trends or abnormalities. Unsupervised techniques encompass clustering, which associates comparable data points, and self-encoder, which can compress data and recognize anomalous patterns by identifying departures from the typical data[10].

Hybrid approaches are a way to significantly enhance malware detection by combining supervised and unsupervised learning techniques. For example, an unsupervised model may initially detect possible abnormalities; then, a supervised model could subsequently categorize these anomalies to determine whether or not they are malware[11]. Accuracy is increased, and false positives are decreased with this combo. Generally, conventional malware detection techniques need help to stay current with novel and changing threats. Machine learning offers a more reliable solution by learning from data and adjusting to new virus trends. While unsupervised techniques aid in detecting novel, unidentified dangers, supervised learning algorithms such as decision trees, random forests, SVMs, and neural networks have demonstrated considerable potential. Combining the two approaches can improve detection performance, which makes machine learning an essential tool for contemporary network security[12].

3 Dataset

Selection of Dataset:

– We have collected 3 types of public domain datasets with traffic flow data they are CCCS-CIC-AndMal-2020, CICAndMal2017 & CTU-13.

– We have chosen to use CTU-13 over other public datasets because it is highly available and has been used quite extensively for many similar research studies in the past.

– The features are the raw attributes in the Netflow data - StartTime, Duration, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes and Label.

Id	Duration(hrs)	# Packets	#NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	167,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,121,077	53GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1
6	2.18	38,764,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,799	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34GB	Virut	1

Fig. 1. 13 different Scenarios of the CTU-13 Dataset

The CTU-13 Dataset was created by CTU University in 2011 and captures real botnet traffic mixed with normal traffic and background traffic. It is made of 13 captures of different botnet samples summarized in 13 bidirectional NetFlow files.
The first scenario uses a malware called Neris. The capture lasted 6.15 hours during which the botnet used HTTP-based C&C channels to send SPAM and perform click fraud. The NetFlow file weighs 368 Mo and is made of bidirectional communications described by 15 features[5].

Metrics: It is necessary to select certain measures to assess the effectiveness of the techniques in order to compare the outcomes of various algorithms. The conventional method is to count the number of false positives, or background communications that are classified as botnets, and false negatives, or botnets that are classified as background communications[5]. To do this, we consider 3 scores as follows: Remember that a low recall is worse than a low precision for our project for detecting malicious software since it indicates that the majority of detected communications are botnets (precision) but the majority of botnet communications go unnoticed (recall).

Naturally, it's simple to have a good recall because all you need to do is mark each transmission as coming from a botnet. In order to ensure that the recall is not too low, the compromise that has been selected is to optimize the f1 score.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{F1-Score} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision+Recal}}$$

Fig. 2. Precision, Recall and F1-Score Formulas

4 Proposed System

In this section, we will be looking towards the final system architecture which is being implemented and is described by the Algorithm and Flowchart.

4.1 Block Diagram

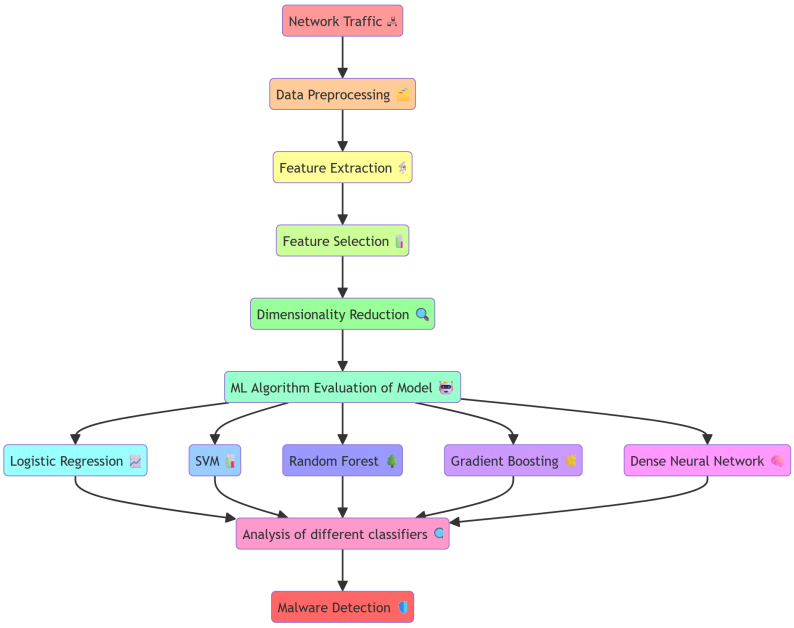

Fig. 3. Block Diagram

1)Network Traffic: The gathering of network traffic data is the first step in the malware detection procedure. Numerous pieces of information are included in this data, such as the number of packets and bytes exchanged, the protocols utilized, source and destination IP addresses, and port numbers. Every network flow's timestamp is also logged. The fundamental input for the complete malware detection procedure is this raw data. We make sure that our next study has a complete dataset from which to discover possible malicious behaviors by recording all pertinent network traffic.

2)Data Preprocessing: Following collection, the raw network traffic data is preprocessed. This is a crucial step since it makes sure the data is clean and well-organized, ready for additional analysis. The process of data cleaning is eliminating any unnecessary, duplicate, or incomplete records that might bias the analysis. In order to standardize the data and guarantee that every characteristic is on the same scale, normalization is also carried out. This stage makes the data more consistent and manageable, which enhances the effectiveness of machine learning models.

3)Feature Extraction: Relevant properties are located and retrieved from the preprocessed data during the feature extraction stage. This entails choosing crucial elements that are most likely to aid in differentiating between malicious and benign network data. These characteristics include things like the length of the connection, the protocols being utilized, and the packet and byte speeds. By concentrating on these useful characteristics, we can improve the accuracy of malware detection by streamlining the data and making it easier for machine learning algorithms.

4)Feature Selection: This block selects a subset of the features extracted in the previous stage. This is done to improve the performance of the machine learning algorithms and to reduce the amount of data that needs to be processed. Different feature selection techniques exist, here we have used Embedded methods, that detect malware specifically focusing on lightweight techniques suitable for resource-constrained embedded devices. These methods prioritize Low computational footprint, Low memory usage, and Real-time detection.

5)Dimensionality Reduction: Techniques for dimensionality reduction are used to further filter the data. The goal of this stage is to minimize the amount of characteristics while maintaining the crucial data required for a thorough analysis. To accomplish this reduction, methods like Principal Component Analysis (PCA) or t-SNE may be applied. The curse of dimensionality, which can make analysis and model training more difficult, is lessened with the use of dimensionality reduction. Additionally, it lowers computational complexity and enhances machine learning models' interpretability and performance.

6)ML Algorithm Evaluation of Model: After implementing the classifiers, their performance needs to be rigorously evaluated. This involves using a set of evaluation metrics to assess how well each model performs in distinguishing between benign and malicious traffic. Common metrics include Accuracy, Precision, Recall, F1 Score, and

the scatter plot. Evaluating the models helps identify any potential shortcomings and ensures that the classifiers are reliable and effective.

7)Different Machine Learning Algorithms: Using different machine learning algorithms is the next step after having a simplified dataset available. These classifiers are algorithms that are meant to learn from the data and forecast the likelihood of dangerous or benign network traffic. Logistic Regression, Support Vector Machines (SVM), Random Forests, Gradient Boosting, and Dense Neural networks are the classifiers we have implemented. Every classifier has advantages and disadvantages, and using a variety of them enables a thorough analysis to choose the best one for our particular use case.

8)Analysis of Different Classifiers: The performance of the different classifiers is examined and contrasted after the assessment. The comparison analysis is essential to determine which model performs the best. We can identify which classifier offers the best accuracy and dependability in malware detection by looking at how each one performs on the assessment metrics. By doing this step, we can be confident that the best tool for our malware detection system is chosen.

9)Malware Detection: The procedure culminates in the real malware detection step, which employs the top-performing classifier found during the analysis stage. This classifier is used to continually assess incoming network data in a real-time monitoring system. Any communication that seems suspicious can be appropriately flagged by the classifier based on assessment and training data. For network security to be maintained and possible attacks to be quickly addressed, real-time detection is crucial.

4.2 Feature Extraction

Motivation

NetFlow data, commonly used for network traffic analysis, presents a significant hurdle: most of the information is categorized. While converting these categories into yes/no (boolean) columns might seem intuitive, it leads to a massive increase in the data matrix size. This expansion can cause memory errors, even when using compressed sparse matrix formats. To overcome this obstacle, researchers have explored alternative approaches, including feature extraction techniques identified through a review of network traffic analysis literature.

Use of time windows

A common approach for analyzing NetFlow data involves summarizing it within time windows. This strategy capitalizes on the "temporal locality behavior" of botnets, meaning their activity often clusters within specific timeframes. Summarization also offers two advantages:

- Reduced Data Size: This makes it easier to handle large datasets and potentially improves processing speed.

– Real-Time Detection: By analyzing summarized data after each time window, we can achieve near real-time botnet detection.

However, defining the optimal time window size (width) and the gap between windows (stride) remains a challenge. Existing research provides various options, often based on experience rather than clear data-driven methods. Here, we opted for a 2-minute window width with a 1-minute stride.

Another hurdle emerged when considering how to group NetFlow entries within a window. Using connection start time resulted in a significant data size increase. To address this, we opted to utilize the communication duration as an additional feature within the window summary.

Extracted features

NetFlow data is a valuable resource for network traffic analysis, but it presents a particular challenge: most of the data is categorical. Converting these categories into numerical values can be problematic because it creates a significant increase in the size of the data matrix. This can lead to memory errors, even when using compressed data formats. To address this challenge, researchers have developed techniques to extract new features from the data. These features are based on statistical summaries of the original categorical features.

4.3 Feature Selection

Embedded methods

Embedded methods offer another approach to feature selection in network traffic analysis. Unlike filter methods that rely solely on feature characteristics, embedded methods leverage the classifier itself during the training process. This allows them to select features based on their contribution to the classifier's performance, essentially using the classifier's "score" (prediction accuracy) as a guide.

Lasso and Ridge logistic regressionLogistic regression is a popular choice for network traffic analysis tasks. However, it can struggle with imbalanced datasets, where one class (e.g., normal traffic) significantly outweighs the other (e.g., botnet traffic). This imbalance can bias the model towards the majority class, leading to poor detection of the minority class.

To address this challenge, we can leverage cross-validation for data balancing. Cross-validation involves splitting the data into training and testing sets multiple times. During training, we can employ techniques like class weighting within the logistic regression model. By using cross-validation, we can evaluate the effectiveness of different weight values and ultimately identify the best configuration for balancing the dataset and achieving optimal performance with logistic regression.

In our analysis, we found that the best F1-score is achieved by a class weight of 0.044 within the logistic regression model (refer to Figure 4). This weight balances botnet data's influence with background data efficiently, making sure that the model accords more attention to rare botnet examples.

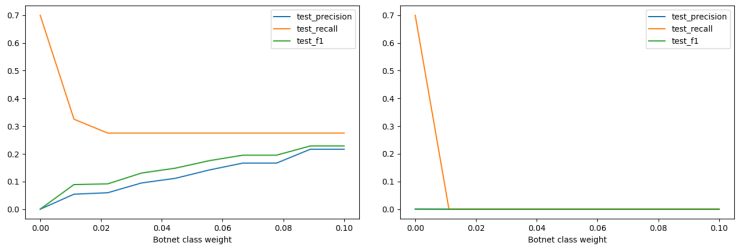

Fig. 4. Shows further improvement on the performance of our model, we will be using L2 regularization (with Ridge) when choosing out features. This method will enable us to identify the most relevant variables and coefficients in our equation model.

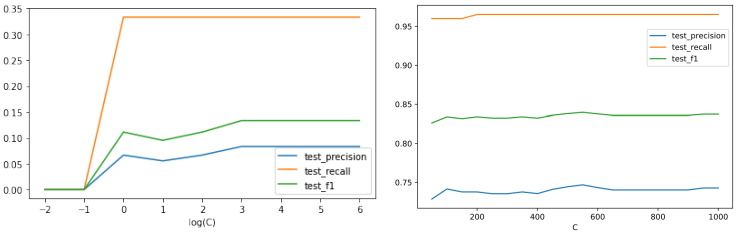

Fig. 5. Illustrates the progression of various scores in relation to the regularization parameter C. The optimal f1 score is achieved when C equals 550, thus necessitating a thorough analysis of the feature coefficients at this specific parameter.

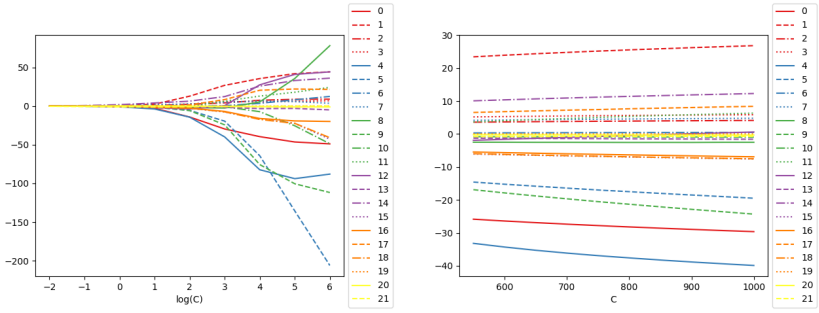

Fig. 6. Feature coefficients wrt log(C), Feature coefficients wrt C (Zoom). It was hard to use L1 regularization efficiently because it has some problems such as convergence issues which hinder implementation. Therefore, L2 regularization, otherwise known as Ridge, is still preferred over others because of increased computational power which enhances feature significance understanding.

4.4 Dimensionality Reduction

Applying methods of dimensionality reduction presents an alternative approach for diminishing the quantity of features within a dataset. Additionally, the visual depiction of the dataset proves to be highly beneficial.

Principal Component Analysis (PCA):

The Principal Component Analysis method results in a set of linearly uncorrected variables and allows the dataset dimension to be reduced.PCA is a statistical technique used to transform a large set of variables into a smaller one that still contains most of the information in the large set. It achieves this by finding a new set of uncorrelated variables, called principal components, which are linear combinations of the original variables.[7].

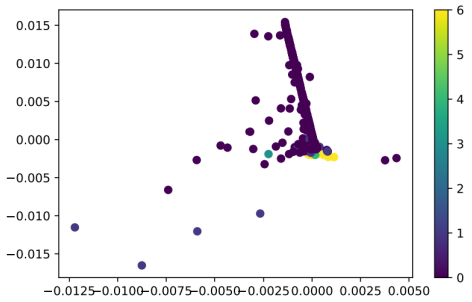

Fig. 7. Principle Component Analysis

The figure shown is the representation of the initial two PCA components with the pre-selected feature set shown in Figure. Six botnets are identified, and specific non-botnet points are hidden. 58 percent of the dataset variation is represented by the first component, the vertical axis, and the second component, the horizontal axis, explains 35 percent of the total variation.

The botnet indicates that it meets at the base of a long, straight line of background points. One can train a K-Nearest Neighbors classifier using this representation.

t-SNE (t-distributed Stochastic Neighbour Embedding):

Another technique to reduce the dataset's dimension to two or three features is t-SNE. The botnets are labeled as 6, which displays the outcome of the dimensionality reduction. The botnet points are dispersed throughout the network, making distinguishing them from real traffic difficult. The algorithm also uses a lot of memory and time.

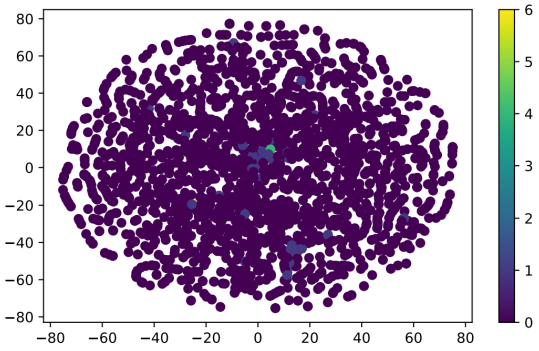

Fig. 8. t-SNE

Therefore, in this Report, the K-nearest neighbor method is no longer considered for detecting botnet activities.

Starting with the flow chart, Data preprocessing, which involves cleaning and preparing raw network data, is where the flowchart begins. Subsequently, the Data Collection stage combines the processed data into a dataset to extract meaningful attributes for Feature Extraction. The next step is Model Selection, where appropriate machine learning algorithms like Random Forest, SVM, Gradient Boosting, Dense Neural Networks, and Logistic Regression are selected. After using the dataset for training, the model's performance is assessed. It is tested on fresh, untested data to ensure the model is accurate and robust. The process ends when the verified model is deployed for real-time malware detection in network traffic.

5 Experimentation and Results

The training of Malware attack detection using network flows with machine learning is done on vs code by creating a virtual environment and installing the required libraries in the same the system has NVIDIA K80 / T4 @2.4 GHz GPU with 16GB RAM.

5.1 Algorithms

Logistic Regression

An algorithm called the Logistic Regression approach classifies the flows by combining characteristics in a linear fashion. To fine-tune the model's parameters, cross-validation is required. The final values that were selected were Weight non-botnet = 0.044 and C = 550.

Random Forest

Several decision tree classifiers are used by the Random Forest method to estimate the class of each input flow. There are one hundred (100) trees in the forest, as chosen.

Support Vector Machine

The Support Vector Machine technique transforms the data space using kernels and then looks for a distinct line to divide the data into two classes. Because cross-validation is required to adjust the model's parameters. The selected alpha parameter for a linear kernel is alpha = 10 raise to 9.

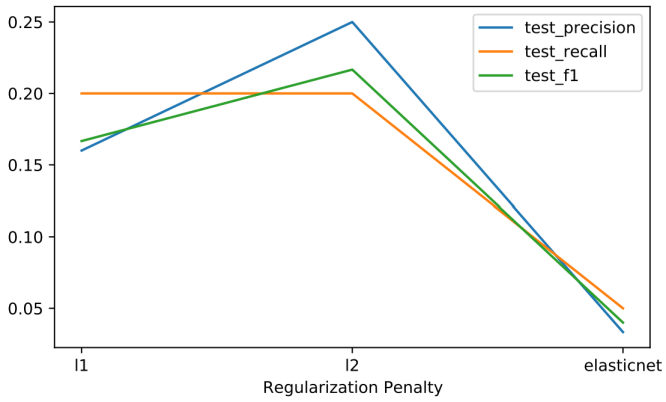

Fig. 9. Determines the most effective regularization technique, cross-validation is used. Based on the findings, a l2 regularization is recommended.

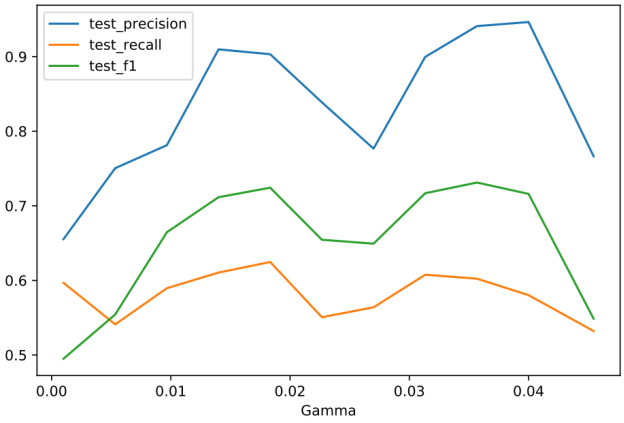

Fig. 10. Displays the outcomes The optimal Y parameter for a Radial Basis Function (RBF) kernel is Y = 0.03567.

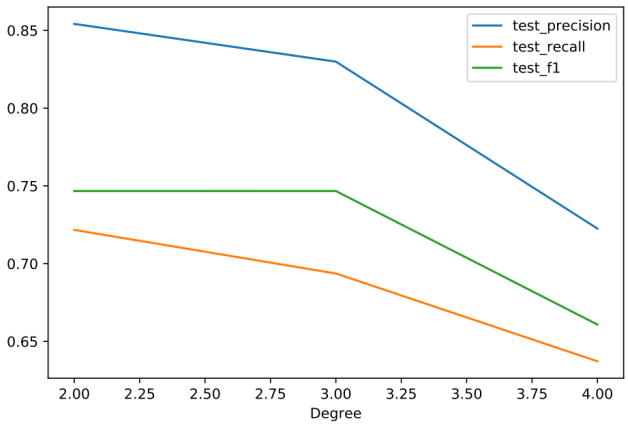

Fig. 11. Illustrates the optimal degree for a polynomial function with a polynomial kernel 2. In conclusion, a polynomial kernel with a degree of two is the optimal set of support vector machine (SVM) parameters for botnet identification in the first scenario of CTU-13.

Gradient Boosting

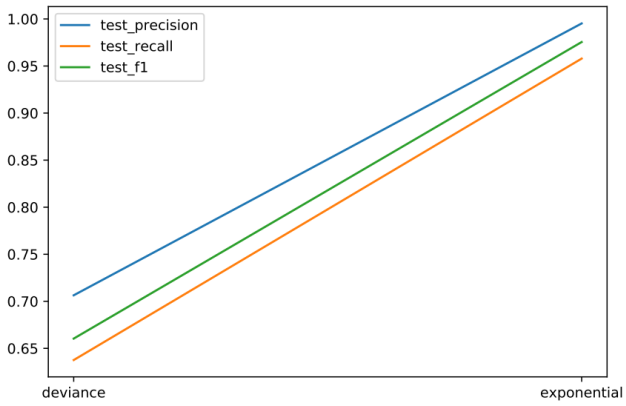

Fig. 12. Shows that an exponential loss performs better than a deviance loss. Using the score from one decision tree to construct a new one, the gradient-boosting algorithm builds new trees with the goal of gradually improving training. The function loss and the maximum depth of the trees (100 is the specified number of trees) are the two primary parameters that need to be adjusted.

Dense Neural Network

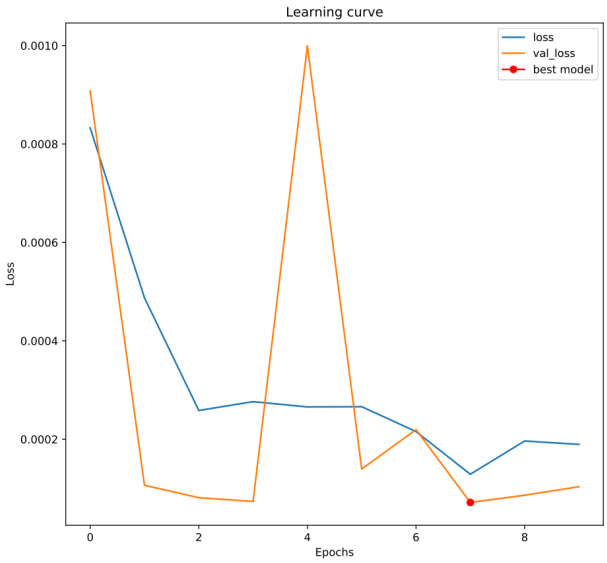

Fig. 13. Learning curve of the Neural Network

Shows Botnet Neris Scenario 1 - Result Summary which demonstrates that with a f1-score of 0.97, Random Forest, Gradient Boosting, and the Dense Neural Network outperform other studied algorithms in the detection of botnets among network traffic. However, both the Support Vector Machine and the Logistic Regression approaches achieve a f1 score of 0.85; the latter with a low recall and the former with a low accuracy. Neural networks are getting more and more popular lately since they function very well with large amounts of data. Here, we evaluate a basic two-layered neural network, which is dense (or completely linked) and includes 256 neurons in its first hidden layer and 128 in its second.
The batch-normalization, no dropout, and ReLU activation function comprise the neural network's parameters (with the exception of the output layer, which uses a sigmoid function). There are 768 non-trainable parameters and 39 681 trainable parameters in the model. The binary cross-entropy loss for the training set and the validation set—which represents 15% of the entire set across 10 epochs is displayed in Figure 3.6 (a batch of 32 is utilized).

5.2 Comparison of algorithms

We train our models using two-thirds of the dataset (randomly selected NetFlows) and test them using the remaining one-third of the dataset in order to compare the approaches[10].

Table Contents

Algorithm: Lists various machine learning algorithms (Logistic Regression, Support Vector Machine, Random Forest, Gradient Boosting, Dense Neural Network).
Training: Shows performance metrics for each algorithm during the training phase.
Test: Shows performance metrics for each algorithm during the testing phase.
For both training and testing, the following metrics are provided for each algorithm:
Prsn: Likely refers to precision, a measure of how accurate the positive predictions are.
Recl: Likely refers to recall, a measure of how many actual positive cases the model correctly identified.
f1-Scr: Likely refers to the F1-score, a harmonic mean of precision and recall, providing a balanced measure of performance.

Algorithm	**Training**			**Test**		
	Prsn	**Recl**	**f1-Scr**	**Prsn**	**Recl**	**f1-Scr**
Logistic Regression	0.75	0.96	0.84	0.75	0.97	0.82
Support Vector Machine	0.96	0.82	0.83	0.91	0.76	0.83
Random Forest	1.0	1.0	1.0	0.99	0.96	0.99
Gradient Boosting	1.0	0.98	1.0	0.97	0.95	0.98
Dense Neural Network	0.91	0.97	0.93	0.95	0.98	0.97

Table 1. Algorithm performance metrics

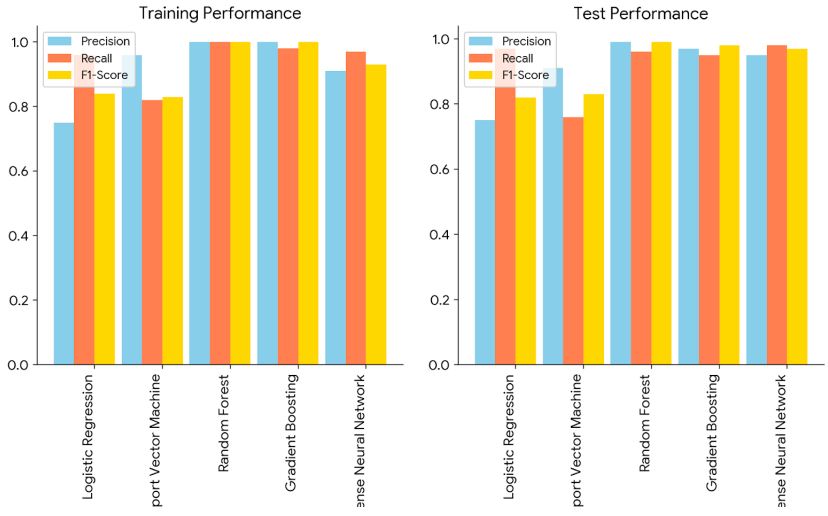

Fig. 14. Algorithm performance metrics

Detection of different Botnets

We test the Random Forest Classifier on all the other situations to check if the outcomes on Scenario 1 of the CTU-13 dataset can be applied to other scenarios[5].

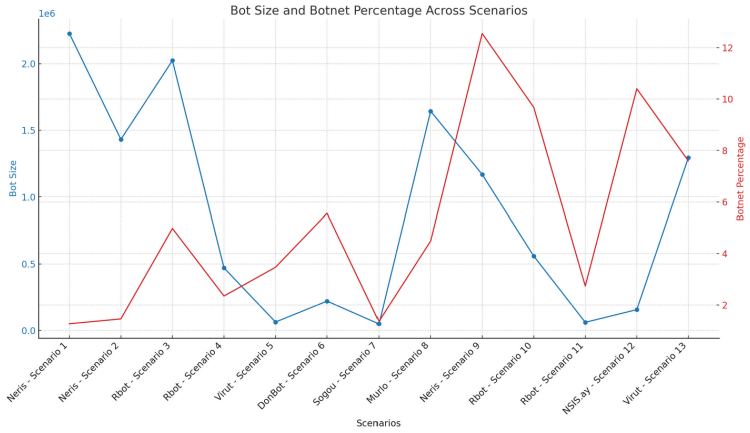

Fig. 15. Bot size and Botnet percentage across various scenarios

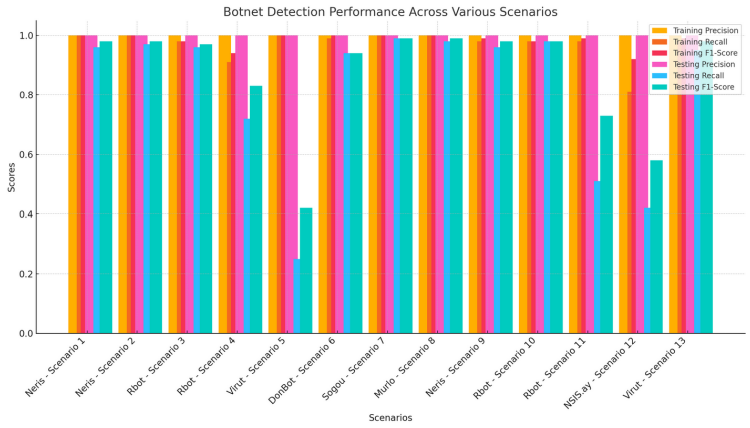

Fig. 16. Botnet detection performance across various scenarios

6 Conclusion

This work endeavors to develop and evaluate algorithms capable of identifying botnet activity inside real-world network traffic, as modeled by Netflow datasets. Relevant characteristics were retrieved following a thorough examination of the data and several reviews of network security literature. After that, their impact was examined through a selection procedure, although none of the features were so subpar as to be excluded from the training portion. Following that, other algorithms were evaluated, including a Dense Neural Network, Random Forest, Support Vector Machine, Gradient Boosting, and Logistic Regression. For eight out of the thirteen cases, the Random Forest Classifier was used to detect botnets, yielding a detection accuracy of over 95 percent of the botnets.

References

[1] Stephen Akandwanaho and Dr Kooblal. "Intelligent Malware Detection Using a Neural Network Ensemble Based on a Hybrid Search Mechanism". In: *The African Journal of Information and Communication* 24 (Dec. 2019), pp. 1–21. DOI: 10.23962/10539/28660.

[2] Muhammad Shoaib Akhtar and Tao Feng. "Evaluation of machine learning algorithms for malware detection". In: *Sensors* 23.2 (2023), p. 946.

[3] Olaniyi Abiodun Ayeni. "A Supervised Machine Learning Algorithm for Detecting Malware". In: *Journal of Internet Technology and Secured Transactions* 10.1 (2022), pp. 764–769.

[4] Dmitri Bekerman et al. "Unknown malware detection using network traffic classification". In: *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2015, pp. 134–142.

[5] Antoine Delplace, Sheryl Hermoso, and Kristofer Anandita. "Cyber attack detection thanks to machine learning algorithms". In: *arXiv preprint arXiv:2001.06309* (2020).

[6] B Kaspersky. *Machine learning methods for malware detection*. 2020.

[7] Ban Mohammed Khammas et al. "Feature selection and machine learning classification for malware detection". In: *Jurnal Teknologi* 77.1 (2015).

[8] Rafał Kozik, Robert Młodzikowski, and Michał Choraś. "Netflow-Based Malware Detection and Data Visualisation System". In: *Computer Information Systems and Industrial Management: 16th IFIP TC8 International Conference, CISIM 2017, Bialystok, Poland, June 16-18, 2017, Proceedings 16*. Springer. 2017, pp. 652–660.

[9] Joël Codina Poquet. "Detection of malware traffic with Netflow". In: *Facultat d'Informàtica de Barcelona* (2017).

[10] Nur Selamat and Fakariah Ali. "Comparison of malware detection techniques using machine learning algorithm". In: *Indonesian Journal of Electrical Engineering and Computer Science* 16 (Oct. 2019), p. 435. DOI: 10.11591/ijeecs.v16.i1.pp435-440.

[11] Matija Stevanovic and Jens Myrup Pedersen. “An efficient flow-based botnet detection using supervised machine learning”. In: *2014 international conference on computing, networking and communications (ICNC)*. IEEE. 2014, pp. 797–801.

[12] Somendra Tripathi, Hari Om Sharan, and CS Raghuvanshi. “Intelligent Data Encryption Classifying Complex Security Breaches Using Machine Learning Technique”. In: *Effective AI, Blockchain, and E-Governance Applications for Knowledge Discovery and Management*. IGI Global, 2023, pp. 143–157.