Using Machine Learning to detect various Botnets

Rasmus G. K. Christiansen[1] **and Bernhard Zosel**[1]

[1]*Reykjavík University, Island, Menntavegur 1 IS-102 Reykjavík*

Compiled November 24, 2023

This paper explores supervised machine learning approaches for the detection of botnets, with a focus on the ISOT HTTP Botnet dataset from 2017. The focus is on exploring the capabilities of different supervised learning methods to improve the accuracy and effectiveness of botnet detection systems. The study evaluates several machine learning approaches, including Decision Trees, Random Forests, XGBoost, and deep learning models like Multi-Layer Perceptron. Special attention is given to feature extraction, preprocessing, and the effects of balancing the highly unbalanced dataset by using techniques like SMOTE. The findings contribute to a deeper understanding of machine learning's potential in enhancing cyber defense mechanisms against the evolving threat of botnets. We find that in both the binary setting for detecting malicious botnet activity but also in multi-class settings to detect the particular type of botnet, straightforward approaches such as Random Forest and XGBoost proved to be the most effective.

GitHub repository with all of the files:
https://github.com/Rodhaaret/T-710-MLCS-Final-project.git

1. KEYWORDS

Botnets, Cyber-Security, Machine Learning, ISOT HTTP Botnet Dataset

2. INTRODUCTION

Computer systems exposed to the internet are constantly threatened by potential attackers, each with their own motivation. These actors could be, hackers, traitors, terrorists, or even governments, seeking to compromise the computer system to example gain availability to protected assets. The attackers will find and take advantage of the computer systems' vulnerabilities to conduct their cyber attacks. Hence, there is an endless loop of developing new and powerful security measurement to combat the increased sophisticated methods these attackers use.

According to the 2020 Threat Landscape Report from the European Union Agency for Cybersecurity (ENISA), some of the most prevalent security threats include malware infections, web-related attacks, phishing, denial of service, spam, and botnets [1]. Among these, botnets emerge as a particularly formidable challenge in the realm of cybersecurity. A botnet is essentially a network of malware-infected devices connected to the Internet. This network comprises two key components: bots, which autonomously execute commands, and botmasters, devices directly controlled by the attacker, responsible for issuing commands to the bots.

The multifaceted nature of botnets makes them a versatile tool for cyber adversaries. They have been leveraged for various malicious activities, primarily falling into three categories within the context of cybersecurity [2]:

- **Spam:** Sending bulk electronic correspondence.
- **Data breach:** capture sensitive information.
- **DDoS:** Distributed Denial of Service attacks

HTTP botnets present a particularly formidable challenge in cybersecurity due to their stealthy nature. The use of HTTP for command and control communications allows these botnets to operate under the radar, as their traffic easily blends with the vast amounts of legitimate HTTP traffic on the internet. This makes recognizing and distinguishing HTTP botnet activities from regular web usage extremely difficult.

Given these huge cyber security challenges, understanding and mitigating the threats posed by botnets is of paramount importance. Given the urgency to develop effective countermeasures, this paper aims to delve into the area of botnet detection. Specifically, we explore how machine learning, with its pattern recognition and anomaly detection capabilities, can be used as a powerful tool to strengthen cybersecurity defences by detecting malicious botnet network traffic and categorising it based on the specific botnet it originates from.

3. BACKGROUND

Many different approaches and methodologies have been proposed over the years for dealing with malicious activities. In this section, we cover the background related to the machine-learning approaches with the purpose of defining what the state-of-the-art is.

In 2017 the ISOT HTTP Botnet Dataset was introduced alongside a novel framework for detecting HTTP-based botnets by Alenazi [3]. He proposed three models: (1) The Domain Mass Detector looks at the features of the DNS queries. (2) The Application Detector's job is to profile the host application by comparing the DNS traffic from legitimate profiles with suspicious ones based on different domain features. (3) The Time Series Detector looks at the timing patterns of the botnets, in which the scheduled communication were found by analyzing inter-queries intervals between the bot itself and the C&C server.[1] They concluded that Random Forest and Decision Tree models outperform models like Naive Bayes Gaussian.

The more recent work by Lopes et al. [13] also attempted to detect botnets based on their network flow behavior. They trained and tested their energy-based flow classifier using the same ISOT HTTP botnet dataset from 2017.

BotCap, which was proposed in 2018 by Gadelrab [4], is a machine-learning approach for detecting botnets using statistical features from the network traffic. They showed that BotCap is capable of detecting infected hosts on a local network, without extensive collection of data features from the infected devices. They generated and used a dataset consisting of two groups. Group (1) consisted of the botnets, `Aryan`, `Ngr`, and `Rxbot`, whose characteristics are that they communicate through the IRC channel. Group (2) consisted of the botnets, `Black Energy`, `Zeus`, and `Vertexnet`, who are known for HTTP communications. They used 9 features they deemed the most relevant from the dataset consisting of both benign and botnets. Then they trained a J48 decision tree and a Support Vector Machine (SVM) using Grid Search Algorithm and a 5-fold cross-validation analysis.

The DroidFusion framework, which was proposed in 2019 by Yerima [5], has shown to be strong at detecting various malicious activities, hence not only for botnet detections. The DroidFusion model used, —*J48 decision tree*, *Forest REP*, *Random Forest-100*, *Random Forest-9*, and *Perceptron* on a test data set extracted using cross-validation on the Malgenome-215 project dataset. Afterward, it was evaluated on 3 different datasets, Drebin-215, McAfee-350, and McAfee-100. They concluded that the DroidFusion framework's key components of using the supervised models and calibration strategy improved the resultant accuracy and precision.

In 2019 Zhou and Pezaros introduced a methodology aimed towards finding Zero-Day intrusions[7][2]. They proposed the idea of using the CICFlowMeter-V3 tool to extract and analyze behavioral metrics from the CIC-AWS-2018 dataset in this case. They showed that most of the novel classifiers had outstanding performance in botnet detection. They also concluded that the Decision Tree model had lower overhead time compared to Random Forest, Naive Bayes, Decision Tree, Neural Networks (MLP), Discriminant Analysis, and K-Neighbors. Hence, the Decision Tree model is suitable for statistical data, and therefore suitable for pattern detection of botnets.

A key challenge for supervised machine learning approaches is the unbalanced nature of the different classes that a lot of the datasets come with. Synthetic Minority Oversampling Technique (SMOTE), which was introduced in 2020 by Gonzalez-Cuautle [8] aims to limit this issue. The SMOTE generates synthetically balanced data and optimally calibrates the parameters of the machine learning classifiers to limit overfitting. The SMOTE is used before training the classifier as it creates the minority- and majority classes. On top of this, they used Principal Component Analysis (PCA) to get the relevant features and Grid Search algorithm to estimate the hyperparameters even further. In conclusion, SMOTE with Grid Search algorithm improved the prediction of malicious activity in highly unbalanced data sets.

[1]A Command and Control (C&C) server is a centralized system used by cybercriminals to remotely manage and control compromised computers or devices within a botnet for malicious activities.

[2]Zero-Day intrusions: refer to cyberattacks that exploit vulnerabilities in software or hardware that are unknown to the vendor and, therefore, have no available patches or fixes.

Around the same time in 2020, Asadi [9] proposed a combination use of optimization algorithms, classic machine learning algorithms, and deep learning to detect botnets. Firstly, he uses the particle swarm optimization (PSO), to get great features for detecting botnets, in combination with the BD-PSO-V voting system to combat the previously discussed challenges. Deep neural network algorithms, support vector machines (SVM), and C4.5 decision trees are also used to identify botnets.

BotStop, which was introduced in 2022 by Alani[10], is one of the newer methods in the machine learning approaches category. BotStop examines both incoming and outgoing packages in a network. This method is using explainable machine learning (ML) algorithms with features extracted from network packets. Compared to previous methods that examine the flow this method looks directly at the package level. Hence, he showed that BotStop is significantly faster at detecting malicious activity compared to the flow-based.

4. HYPOTHESIS

From this we defined two hypotheses to test:

1 : Training a binary classifier to distinguish traffic from benign applications from malicious traffic originating from botnets

2 : Training a more sophisticated multi-class machine learning classifier to also detect the botnet type that the malicious network flows are originating

In both approaches, the classification is based on the network flow behavior and the 2017 ISOT HTTP Botnet Dataset is used to train and evaluate the classifiers.

5. METHODOLOGY

We divide into the more classical machine-learning approaches in a supervised manner: *Naive Bayes*, *Logistic Regression*, *Decision Trees*, *Random Forests* and *XGBoost*. On top of this, we train deep learning models such as *Multi-Layer Perceptron* and a deep neural network with Keras to see how it performs against the more classical methods in terms of generalization on new unseen data. The following subsection dives into how the training and testing are set up.

The Dataset: The ISOT HTTP Botnet Dataset from 2017, based on the work from Alenazi A, is one of the more recent realistic datasets for this specific security threat [12]. It consists of two parts: (1) a botnet dataset generated by capturing malicious DNS traffic only. (2) benign traffic gained after capturing legitimate traffic generated by different applications such as antivirus, online chat, and browser applications to name a few. The data has been collected over multiple days running in a virtual environment, which can be seen depicted in Figure 1. The malicious data has been generated using in total of 9 different types of botnets. The IP/name of the bots within the dataset is given as:

- 192.168.50.14 for `zyklon.botnet.isot`
- 192.168.50.15 for `blue.botnet.isot`
- 192.168.50.16 for `liphyra.botnet.isot`
- 192.168.50.17 for `betabot.botnet.isot`

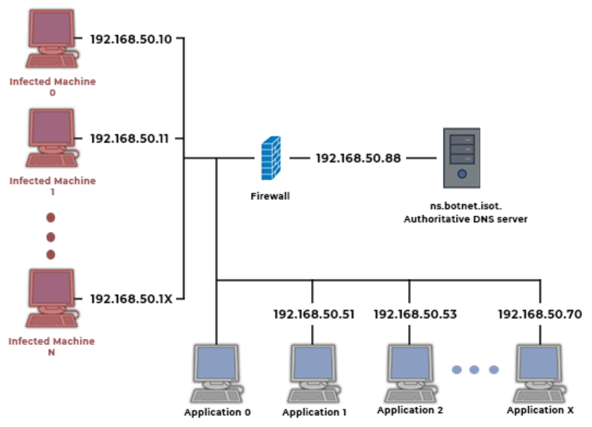


Fig. 1. The virtual environment setup for generating the ISOT HTTP Botnet Dataset.[1]

- 192.168.50. 18 for `blackout.botnet.isot`
- 192.168.50.30 for `citadel.botnet.isot`
- 192.168.50.31 for `citadel.botnet.isot`
- 192.168.50.32 for `be.botnet.isot (Black energy)`
- 192.168.50.34 for `zeus.botnet.isot`.

The generated network data has been separated into two parts - 3 .pcap files for network traffic containing only benign traffic and 5 .pcap files containing data with malicious traffic from the botnets.

Feature Extraction & Labeling: Firstly, the CICFlowMeter tool [6] is used to extract the network flows from each .pcap file. The tool generates a csv file with 84 flow-based metrics, for example the flow duration and the number of packets sent and received. We manually remove the Flow ID, Source IP and Port, Destination IP and Port, and Timestamp features, as they are too specific for each flow and would not generalize well, e.g. when new botnets were used.

After the feature extraction, the network flows are being labeled. All flows originating from the application datasets are labeled as 'Benign', while in a first step, every flow originating from one of the 9 malicious ip addresses in the botnet dataset is just labeled as 'Malicious'. In a second step, each malicious flow is labeled according to its particular botnet, from which the flow originates from. Figure 2 shows an example network flow that has been preprocessed with CICFlowMeter and has been labeled. For better understanding, we have also included the flow ID as well as the source and destination IP addresses and ports, although these will be removed during training.

Initial Data analysis: When analysing the distribution of the different botnets in the dataset files, it is evident that the dataset is heavily unbalanced. `citadel2.botnet.isot` is present 24.56% of the time, while the `be(BlackEnergy).botnet.isot` is only present 2.93% of the time. The full distribution can be seen in Figure 3. Not dealing with the unbalanced nature of the dataset could lead to classifiers with bad recall ability and in general bad generalization, which was the case for Alenazi [3] using this exact same dataset. To investigate this we generate two setups. (1) the 8 different botnet are categorized into a subset for the *majority classes* that being *Citadel*, *Zeus*, and *Citadel2* and a *minority classes* that being *Blackout*, *Blue*, *Liphyra*, *Black Energy* and *Zyklon*. This means that the classifiers are being trained with the "balance" feature enabled to counter this. This means that the classes in the minority group will have a bigger weight compared to the once found in the majority group. (2) second options is to simply keep the dataset as it it without balancing it to see the affect of it.

Preprocessing: Both the balanced and the unbalanced data set need some further preprocessing. After invalid entries have been removed, the *protocol* column must be one-hot-encoded, since the type of protocol is categorical and should not be interpreted as a continuous value. While decision trees and random forests can handle unscaled data, in the other approaches the features are standardised using *StandardScaler* from sklearn.

Train & test data: The data has been split 70% for training phase and 30% for testing phase. Each of the classifiers are being trained with multiple different parameters using Grid Search with 5-fold cross validation. For each of the classifiers used, some assumptions are made, which are described in the following section.

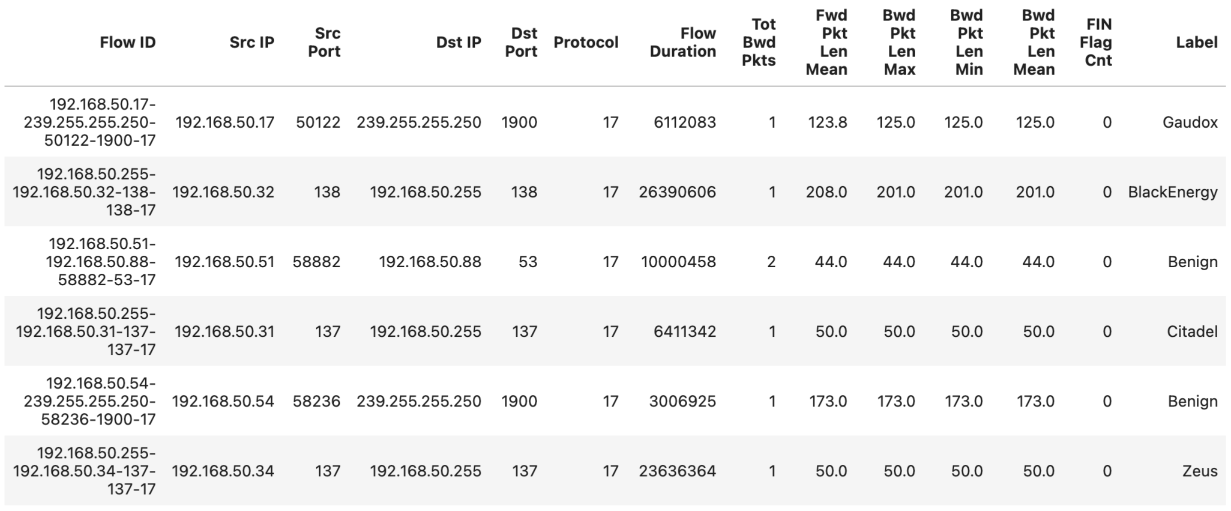
Naive Bayer Classifier: The Naive Bayes classifier is known for its efficiency with large feature sets. However, it's important to note that this model assumes feature independence, an assumption which does not hold in our botnet detection context due to the interrelated nature of network flow data. Despite this, the simplicity and efficiency of Naive Bayes can provide initial insights into our dataset, at least for the binary case.

Logistic Regression: Logistic Regression is known for its effectiveness in binary classification tasks. This model is particularly suited for situations where the relationship between the independent variables and the log-odds of the dependent variable is linear. However, in the context of our botnet detection task, the linearity assumption of Logistic Regression might not fully capture the complex relationships between features. Despite this, the model's interpretability made it a valuable tool for our initial analysis.

Decision Tree Classifier: For decision trees, the maximum depth is the most important parameter to fine-tune. Too deep trees tend to overfit, while too shallow trees do not explain the data good enough and underfit.

Decision Trees offer the distinct advantage of not requiring data standardization, making them well-suited for our diverse range of network traffic data. A crucial aspect of working with Decision Trees in our context was fine-tuning the max_depth parameter, which is essential to balance between underfitting and overfitting. While a deeper tree might capture more data complexity, it also risks overfitting to our specific dataset. Therefore, careful optimization of max_depth was a key focus to ensure the model's generalizability and effectiveness in identifying various botnet activities. A big advantage of decision trees is the fact that we can easily extend them for multi-class classification tasks, which enables the detection of the particular botnet that malicious traffic orgininated from.

Random Forest Classifier: Random Forests are a powerful ensemble method that combine multiple Decision Trees to improve classification accuracy. This approach is particularly effective in our multi-class botnet detection scenario, as it can handle the complexities and nuances of various types of botnet traffic. Random Forests inherently manage overfitting better than



Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Flow Duration	Tot Bwd Pkts	Fwd Pkt Len Mean	Bwd Pkt Len Max	Bwd Pkt Len Min	Bwd Pkt Len Mean	FIN Flag Cnt	Label
192.168.50.17-239.255.255.250-50122-1900-17	192.168.50.17	50122	239.255.255.250	1900	17	6112083	1	123.8	125.0	125.0	125.0	0	Gaudox
192.168.50.255-192.168.50.32-138-138-17	192.168.50.32	138	192.168.50.255	138	17	26390606	1	208.0	201.0	201.0	201.0	0	BlackEnergy
192.168.50.51-192.168.50.88-58882-53-17	192.168.50.51	58882	192.168.50.88	53	17	10000458	2	44.0	44.0	44.0	44.0	0	Benign
192.168.50.255-192.168.50.31-137-137-17	192.168.50.31	137	192.168.50.255	137	17	6411342	1	50.0	50.0	50.0	50.0	0	Citadel
192.168.50.54-239.255.255.250-58236-1900-17	192.168.50.54	58236	239.255.255.250	1900	17	3006925	1	173.0	173.0	173.0	173.0	0	Benign
192.168.50.255-192.168.50.34-137-137-17	192.168.50.34	137	192.168.50.255	137	17	23636364	1	50.0	50.0	50.0	50.0	0	Zeus

Fig. 2. Example for labeled network flows (note: not all features are shown)

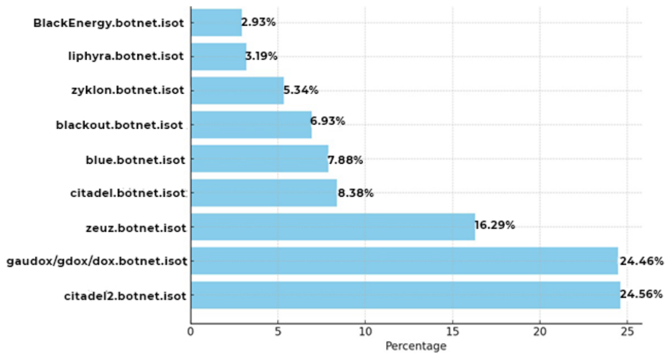


Fig. 3. Distribution of botnet traffic samples grouped by class in the ISOT HTTP Botnet dataset.

individual Decision Trees, making them more reliable for our quite complex dataset. Additionally, like Decision Trees, Random Forests do not require the standardization of data, which simplifies the preprocessing steps.

A critical factor in the effectiveness of Random Forests is the optimization of parameters, particularly the number of trees and their maximum depth. Fine-tuning these parameters was essential in our study to balance the model's ability to capture sufficient complexity without overfitting.

Multi-Layer Perceptron: We employed a Multi-Layer Perceptron (MLP), a type of neural network, known for its proficiency in handling complex, non-linear patterns in data, which is crucial for our multi-class botnet detection task. MLPs are particularly effective in scenarios like ours, where the relationships between input features and output classes is not easily separable by linear methods.

One of the key aspects of our MLP implementation was the architecture of the hidden layers. We experimented with different configurations. Finally, we decided to test a model with one hidden layer with 100 neurons, and another model with two subsequent layers with 50 neurons each. The choice of hidden layers and neurons defines the learning capacity of the MLP. With too few neurons, the model might not capture all the complexities of the data, whereas too many could lead to overfitting. Additionally, we tried out different activation functions, learning rates (adaptive vs. constant) and modified the alpha parameter.

Neural Network (using Keras): Neural Networks with Keras offer greater flexibility compared to the Multi-Layer Perceptron. This advanced approach provided us with the ability to design a more complex network architecture, with the possibility of Dropout Layers, Recurrent Layers and more. However, this expanded flexibility also introduced a higher level of complexity in fine-tuning the model. The process of optimizing the architecture and hyperparameters became more intricate. We have found that simple levels are already more than sufficient to achieve decent performance and that it is difficult to improve them with additional, more complex levels.

XGBoost: Another more sophisticated approach that we incorporated, is XGBoost (eXtreme Gradient Boosting), an advanced implementation of gradient boosting machines, renowned for its high efficiency, flexibility, and performance. XGBoost is a decision-tree-based ensemble machine learning algorithm that uses a gradient boosting framework, making it particularly effective for complex classification tasks like our multi-class botnet detection.

In XGBoost, models are built sequentially, with each new model correcting the errors of the previous ones, thus leading to a strong predictive performance. Its ability to handle various types of data, robustness to missing values, and advanced regularization features to prevent overfitting made XGBoost especially suitable for our dataset's complexity. The tuning of its hyperparameters, including learning rate, tree depth, and number of estimators, was crucial, offering a balance between model complexity and overfitting.

Evaluation metrics: In binary classification, we mainly focused on accuracy, while in multi-class classification, the precision and recall of each class are very important to see the performance of the minority classes, which do not affect the overall accuracy for such a large measure. For better illustration, we also used confusion matrices.

Used Environment: The code was mainly developed in Python (v3.9.7) using Jupyter Notebooks, which were chosen for their flexibility in experimenting with different algorithms and the ease of parameter adjustment. For the machine learning classifiers we used scikit-learn (v1.3.0), Pandas (v2.0.3) for the classical approaches and Keras (v.2.14.0) for the neural network method. All of the files, code and results can be found in the following GitHub repository [1]

The code has been executed on a MacBook Pro, 2019, with an 8-Core Intel Core i9, Intel UHD Graphics 630, and 32 GB 2667 MHz DDR4 memory.

6. RESULTS

Binary Classification: Table 1 below shows the accuracy for each of the best performing configurations for the different classifiers in the binary setting where they simply have to classify if network flows are *Benign* or *Malicious*. The optimal parameters that have been discovered by the grid search are displayed in a separate column.

Classifier	Parameters	Accuracy
Naive Bayes	var_smoothing=1e-07	93.54%
Logistic Regression	C=10, max_iter=1000	97.49%
Decision Tree	max_depth=10, min_samples_split=2, criterion=entropy, min_samples_leaf=1	99.39%
Random Forest	citerion=entropy, n_estimators=200, max_depth=30	99.46%
MLPClassifier I	hidden_layer_sizes=(100,), learning_rate=constant, alpha=0.0001	98.99%
MLPClassifier II	hidden_layer_sizes=(50, 50), learning_rate=adaptive, alpha=0.001	99.10%

Table 1. Accuracy of binary classification for different classifiers

The best classifier for determining whether a network flow is benign or malicious is the Random Forest, which slightly outperforms MLP and Decision Tree Classifiers. However, if runtime plays a bigger role, than the decision tree is quite a good choice, since it is only slightly worse than the random forest but with better computation time.

Multi-Class Classification: Secondly, our classification is now extended to classify benign traffic and traffic from eight different botnet classes, with the two Citadel botnets grouped as a single category. We employed only the most effective classifiers from the binary classification setting, including various neural network architectures. However, it was observed that random forests consistently outperformed those models. The more intricate neural network designs did not yield any notable enhancements in performance. However, XGBoost demonstrated substantial efficacy, slightly surpassing the performance of random forests. The different accuracies of these models are detailed in Table 2, where XGBoost emerges as the superior classifier, consequently, we focus our further analysis on that model.

Classifier	Accuracy
Decision Tree	80.47%
Decision Tree (balanced)	68.59%
Decision Tree (balanced with SMOTE)	78.66%
Random Forest	81.18%
Random Forest (balanced with SMOTE)	79.57%
XGBoost	81.68%
XGBoost (balanced with SMOTE)	80.19 %
Keras Model	73.98%

Table 2. Accuracy of binary classification

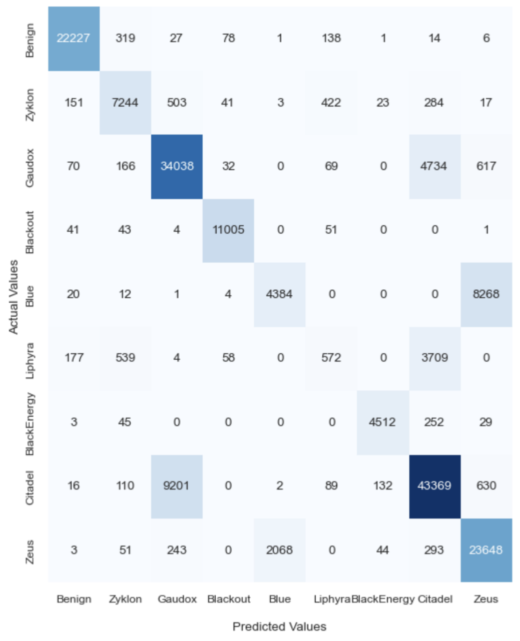


Fig. 4. Confusion Matrix of XGBoost Classifier

Figure 4 presents the confusion matrix generated by the best-performing classifier, XGBoost, and in conjunction with the classification report in Table 3, it reveals that some classes have notably low precision and recall. Especially the Liphyra botnet notably suffers from just 43% precision and 11% recall, a significant concern given its status as one of the dataset's least frequent classes.

Despite the initial promising results from other papers achieved by balancing the dataset, this approach did not lead to an improvement in overall performance, but actually led to a slight decline in most of the key metrics, including accuracy. We have tried many different upsampling and downsampling techniques such as SMOTE without achieving any improvement. However, the methodology involving exclusive upsampling of minority classes did yield partly positive results, after experimenting with a variety of upsampling ratios. While the most problematic underrepresented classes such as Liphyra and Blue improve significantly, this is at the expense of the more common classes such as Citadel.

Class	Support	Precision	Recall
Citadel	28.97%	82% (75%)	81% (78%)
Gaudox	21.49%	77% (77%)	86% (86%)
Zeus	14.25%	71% (73%)	90% (77%)
Benign	12.34%	98% (98%)	97% (97%)
Blue	6.86%	68% (60%)	35% (57%)
Blackout	6.03%	98% (99%)	99% (99%)
Zyklon	4.70%	85% (90%)	83% (84%)
Liphyra	2.74%	43% (63%)	11% (37%)
BlackEnergy	2.62%	96% (97%)	93% (98%)

Table 3. Classification Report for XGBoost Classifier (the percentages in brackets for precision and recall refer to the upsampled dataset using SMOTE)

7. DISCUSSION

The performance of the algorithms In the conducted study, a comprehensive range of algorithms were evaluated, including the Naive Bayes Classifier, Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Multi-Layer Perceptron, a more advanced Neural Network implemented using Keras, and XGBoost. These algorithms were extensively tested in both binary and multi-class classification settings. Within the binary classification setting, the Random Forest algorithm exhibited the most proficient performance, while in the multi-class case, XGBoost slightly outperformed the Random Forest.

Balancing the Dataset In our analysis, we identified three pairs of classes within our dataset that the model consistently struggles to distinguish:

- Citadel and Liphyra
- Citadel and Gaudox
- Zeus and Blue

This confusion between those class combinations poses a significant challenge to achieving optimal model performance. Balancing only helped to a limited extent and required a lot of fine-tuning. In the end, we chose an approach that only upsampled the minority classes, however, only to a limited extend (upsampling all classes to the same extent has significantly worsened all evaluation metrics). While this significantly improves the precision and recall of the Minority Classes (especially for the problematic Blue and Liphyra classes), it also comes with sacrifices for the Citadel class in terms of precision and recall and for the Zeus class in terms of recall. Depending on the use case, however, this compromise may be worthwhile and in future work we could concentrate even more on optimising the SMOTE method. However, it is important to ensure that generalization is still retained and the classification is not overfit too specifically to the current distribution of botnet classes.

The ISOT HTTP Botnet Dataset The dataset is great for evaluating approaches and has been used in numerous papers throughout the years. The dataset is from 2017, which is a bit old for anything that should meet industry standards, as more advanced botnets are in the attackers' hands. However, as of writing this paper, no newer dataset regarding only botnets has been published. Newer papers from 2022 and on forward tend to use the *Bot-IoT Dataset* from UNSW Canberra at ADFA [14]. This dataset has been developed with the key intention of giving a more realistic dataset for a network of multiple IoT-connected devices. Our methodology is also applicable to this dataset and could lead to interesting findings.

Usage in real-life application While it is seen that these approaches can gain an accuracy of over 95%, there are some limitations for real-life applications. These Traffic-flow approaches require the collection of live network data that is then processed to detect if a botnet is present or not. Hence, the algorithms will not work fully as a real-time monitoring approach due to this collection of data streams that leads to a delay in its ability to detect botnets. This could potentially mean that a botnet can do harm before the system is even able to detect it, which is of course not ideal in real-time applications. Either a completely different approach is needed or a more sophisticated extraction of features that even though it has to be processed before is so fast that it can be neglected.

Future work For this paper, we strictly focused on the supervised learning approaches, which indeed means label data is needed. This leads to the common problem of needing a proper dataset, which is often very time-consuming to generate and in specific use cases not even feasible for real-life applications. However, in the scheme of a newer and better method emerges methods that are either semi-supervised learning approaches or completely unsupervised learning approaches that are more suitable for real-time application. Analyzing these approaches and comparing them against the approaches used in this paper could lead to interesting discoveries for future botnet detection implementation and which are more applicable for real-life applications.

8. CONCLUSION

This study has extensively explored the application of various supervised machine-learning techniques for the detection of botnets using the 2017 ISOT HTTP Botnet Dataset. Our research demonstrates that while traditional methods like Naive Bayes, Logistic Regression, and Decision Trees provide decent results in the binary setting to distinguish benign from malicious traffic, for the multi-class case, more advanced algorithms such as Random Forests, XGBoost, and certain deep learning models offer significant improvements.

Among the various methods evaluated, Random Forests and XGBoost emerged as the most effective, striking a balance between accuracy and computational efficiency. However, it was also observed that in scenarios where computational resources are limited, simpler models like Decision Trees can still be viable options.

9. COLLABORATION

The cooperation within the team worked flawlessly. The majority of the time we met in person. We tried to solve the preprocessing together, and for the classification, each of us tried out different classifiers and tested them with different parameters. For the report, Rasmus did most of the first half (introduction, background, methodology), while Bernhard focused on the presentation of the results and their discussion.

10. REFERENCES

1. ENISA. Threat Landscape Report 2020. Available online: (accessed on 16 November 2023).Link

2. Alex Medeiros Araujo, Anderson Bergamini de Neira, Michele Nogueira, Autonomous machine learning for early bot detection in the internet of things, Digital Communications and Networks,2022, Link

3. Alenazi, A.; Traore, I.; Ganame, K.; Woungang, I. Holistic Model for HTTP Botnet Detection Based on DNS Traffic Analysis. In Proceedings of the International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, Vancouver, BC, Canada, 26 October 2017; Springer: Cham, Switzerland, 2017. Link

4. Gadelrab, M.S.; ElSheikh, M.; Ghoneim, M.A.; Rashwan, M. BotCap: Machine learning approach for botnet detection based on statistical features. Int. J. Commun. Netw. Inf. Secur. 2018, 10, 563–579. Link

5. Yerima, S.Y.; Sezer, S. DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. IEEE Trans. Cybern. 2019, 49, 453–466. Link

6. Behavior-Centric Cybersecurity Center, (BCCC) CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) Link *we used this Link*

7. Zhou, Q.; Pezaros, D. Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection—An Analysis on CIC-AWS-2018 dataset. arXiv 2019, arXiv:1905.03685. Link

8. Gonzalez-Cuautle, D.; Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Portillo-Portillo, J.; Olivares-Mercado, J.; Perez-Meana, H.M.; Sandoval-Orozco, A.L. Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusion-detection-system datasets. Appl. Sci. 2020, 10, 794. Link

9. Asadi, M., Jamali, M. A. J., Parsa, S., & Majidnezhad, V. (2020). Detecting botnet by using particle swarm optimization algorithm based on voting system. Future Generation Computer Systems, 107, 95-111 Link

10. Alani, M. M. (2022). BotStop: Packet-based efficient and explainable IoT botnet detection using machine learning. Computer Communications, 193, 53-62 Link

11. Huancayo Ramos, K.S.; Sotelo Monge, M.A.; Maestre Vidal, J. Benchmark-Based Reference Model for Evaluating Botnet Detection Tools Driven by Traffic-Flow Analytics. Sensors 2020, 20, 4501. Link

12. Alenazi A., Traore I., Ganame K., Woungang I. (2017) Holistic Model for HTTP Botnet Detection Based on DNS Traffic Analysis. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham: Link

13. Lopes, D. A., Marotta, M. A., Ladeira, M., & Gondim, J. J. (2022, June). Botnet detection based on network flow analysis using inverse statistics. In 2022 17th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-6). IEEE. Link

14. Koroniotis, Nickolaos, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset." Future Generation Computer Systems 100 (2019): 779-796. Link