IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 20, 2021, accepted August 4, 2021, date of publication August 26, 2021, date of current version September 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3108222

Efficient Detection of Botnet Traffic by Features Selection and Decision Trees

JAVIER VELASCO-MATA[1], VÍCTOR GONZÁLEZ-CASTRO[1], EDUARDO FIDALGO FERNÁNDEZ[1], AND ENRIQUE ALEGRE[1,2]

[1]Department of Electrical, Systems and Automation Engineering, Universidad de León, 24071 León, Spain
[2]Spanish National Cybersecurity Institute (INCIBE), 24005 León, Spain

Corresponding author: Javier Velasco-Mata (javier.velasco@unileon.es)

This work was supported in part by the Framework Agreement between the University of León and Spanish National Cybersecurity Institute (INCIBE) under Addendum 01, and in part by the Formación de Profesorado Universitario (FPU), Spanish Government, under Grant FPU18/05804.

ABSTRACT Botnets are one of the online threats with the most significant presence, causing billionaire losses to global economies. Nowadays, the increasing number of devices connected to the Internet makes it necessary to analyze extensive network traffic data. In this work, we focus on increasing the performance of botnet traffic classification by selecting those features that further increase the detection rate. For this purpose, we use two feature selection techniques, i.e., Information Gain and Gini Importance, which led to three pre-selected subsets of five, six and seven features. Then, we evaluate the three feature subsets and three models, i.e., Decision Tree, Random Forest and k-Nearest Neighbors. To test the performance of the three feature vectors and the three models, we generate two datasets based on the CTU-13 dataset, namely QB-CTU13 and EQB-CTU13. Finally, we measure the performance as the macro averaged F1 score over the computational time required to classify a sample. The results show that the highest performance is achieved by Decision Trees using a five feature set, which obtained a mean F1 score of 85% classifying each sample in an average time of 0.78 microseconds.

INDEX TERMS Computational and artificial intelligence, computer applications, decision trees, machine learning algorithms, optimization methods.

I. INTRODUCTION

One of the main current cyber-threats are botnets: networks of compromised and remote-controlled devices. These networks are dangerous because they can perform massive cyber-attacks, from Distributed Denial-of-Service (DDoS) or spam campaigns [1] to disrupting critical structures such as power distribution grids [2]. In 2018, the average annual cost for a company due to incidents related to botnets was $390,752 [3]. In 2019, the FBI's Internet Crime Complaint Center (IC3) recorded more than $3.5 B in individual, and companies losses related to cybercrime [4]. Moreover, the 2020 report on botnets from the European Union Agency for Cybersecurity (ENISA) [5] counted that 7.7 million IoT devices are connected every day to the Internet, increasing the attack surface for malware infections [6], [7]. Besides, it informed on an increase on botnet controlling servers of 71.5% concerning the previous year, which raises the need to develop specific countermeasures against botnets.

Traditionally, network traffic generated by malware has been detected with signature-based solutions such as Snort [8], or Suricata [9]. Although these methods are highly efficient once a proper malware trait is obtained, they have several downsides. The first problem is that it requires a previous identification of the malware program [10]. Second, it is necessary to perform a deep analysis of the malicious code to obtain the signature, which requires time and resources [11]. Third, recent malware programs have started to use encryption in their communications [12], thus making the signature recognition systems based on Deep Packet Inspection (DPI) ineffective. DPI systems inspect the packets' payloads to determine if certain string patterns can be observed.

The disadvantages of signature-based methods have been overcome with behavior-based detectors, i.e., algorithms that classify traffic according to communication patterns [13]. In our previous work [14], we focused on determining

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Ching Ying.

the bases of a behavior-based botnet detector. Specifically, we created a new botnet dataset and using it to measure the performance of four classifiers: Decision Tree (DT), k-Nearest Neighbors (k-NN), Support Vector Machine (SVM) and Naïve Bayes (NB). We used of 13 traffic features and concluded that DT and k-NN achieved the highest detection scores. Besides, recent works have also confirmed the advantage of DT-based algorithms to classify network traffic, as well as k-NN [15]–[19]. For that reason, in this work, we have selected DT, Random Forest (RF) and k-NN as candidate models to obtain an optimal botnet detection scheme.

While we found multiple academic works that employ feature selection to improve the detection scores of the classifiers [19]–[28], we noticed a lack of research on the performance of the detectors after being trained, i.e., on the effects on the speed of the model for classifying samples. To give insight into this area, we took the feature set we used in [14] and determined the most relevant features. In this process, we used Gini Importance and Information Gain and obtained three feature subsets. Then, we contributed to this research line by measuring the performance of these three subsets. Note that by *performance*, we refer to the relation between the achieved F1 score and the computational time required by the feature subsets for the task of classifying a sample.

The other contribution we present in this work is the development of two datasets based on the well-known CTU-13 database [29]. The main problem of CTU-13 is the imbalance between its classes, so we have proposed the Quasi-Balanced-CTU13 (QB-CTU13) as a selection of samples from the different classes of CTU-13 while preserving the less represented ones. We also added traffic from regular sources (i.e., not botnets) to make a more updated normal class. Additionally, we included traces of three modern and challenging botnets to expand this dataset, creating the Extended QB-CTU13 (EQB-CTU13). The details of the datasets are explained in subsection IV-A. The graphical abstract of Fig. 1 depicts how these datasets have been used, along with the experimentation.

The structure of this work is described as follows. In Section II we review the literature related to botnet detection and feature selection. In Section III the used methods are described, while our experiments are detailed in Section IV. Finally, the results are presented and discussed in Section V, and the conclusions are drawn in Section VI.

II. BACKGROUND

A. BOTNET DETECTION

Early methods for botnet detection such as BotMiner [30] or BotSniffer [31] relied on statistical algorithms to detect botnet traffic. This was possible by taking advantage of the time-space correlations of the traffic generated by the malware programs, as opposed to the traffic produced by real humans, which is more random.

Reviewing the last publications on botnet detection, it can be observed that Machine Learning (ML) algorithms are the most common when the goal is to build multi-class traffic classifiers. Among all of them, models based on Decision Trees (DT) usually achieve the best results. For example, in the work of Gadelrab *et al.* [32], Support Vector Machine (SVM) and DT were compared to select the best classifier for a botnet detector implementation (BoTCap). DT achieved the highest performances (i.e., an F1 score of 95%) over a self-constructed dataset with traces of six botnets: Aryan, Ngr, Rxbot, Blackenergy, Zeus and Vertexnet. Moreover, DT-based algorithms also showed high performance on network attacks datasets, like the CICIDS2017 dataset [33], which collects online attacks such as DDoS or malware infections. This dataset was also used in the work of McKay *et al.* [34] to construct three training sets and two test sets, employed to compare six algorithms: DT, Random Forest (RF), k-Nearest Neighbors (k-NN), Naïve Bayes (NB), Multi-Layer Perceptron (MLP) and One Rule (OneR). The results showed that DT achieved the best F1 score (i.e., 99%) using the 78 features of the CICIDS2017 dataset, and the authors left feature selection as future work.

Another related field where DT based models are used is detecting botnet accounts, i.e., automated users on social networks that propagate spam. Working on this problem, Chu *et al.* [35] classified a collection of 500,000 accounts of Twitter using an RF into three classes: human, bot and cyborg. The model obtained a True Detection Rate of 96%, where *human* was the least misclassified class, with only 1.4% of human accounts wrongly classified. Machine Learning algorithms have also been used in other aspects to detect bot activity, such as over the logs of SSH sessions to identify malicious use of commands [36].

On the other hand, classifiers based on Deep Learning (DL) were recently proposed for binary classification in botnet detection. For example, the work of Maeda *et al.* [37] used a Deep Neural Network (DNN) to identify the botnet traffic traces. Maeda *et al.* reported an accuracy of 99.2% in a dataset built by joining data from the CTU-13 dataset [29], the ISOT dataset [38] and self-captured not-malicious traffic. One of the downsides of detectors based on ML and DL is the possibility of adversarial attacks, when the malicious program output is specially designed to mimic the features of a non-malicious one [39]. Therefore, another benefit of DL-based approaches is the use of Generative Adversarial Networks (GAN), which generate 'fake' samples to be mixed within the dataset, so the trained detector is more resilient against crafted attacks. Yin *et al.* [40] proposed a framework to improve botnet detectors by using a GAN called Bot-GAN. They used the ISCX 2014 dataset [23] divided into 491, 381 samples for training and 348, 452 samples for testing. They found that, while the F1 score of the detector was 68.51% when no fake samples were used, they achieved a peak F1 score of 70.59% if 500 fake samples were mixed with the training data.

To the best of our knowledge, the most used dataset to test botnet detectors is the CTU-13, either pure or with

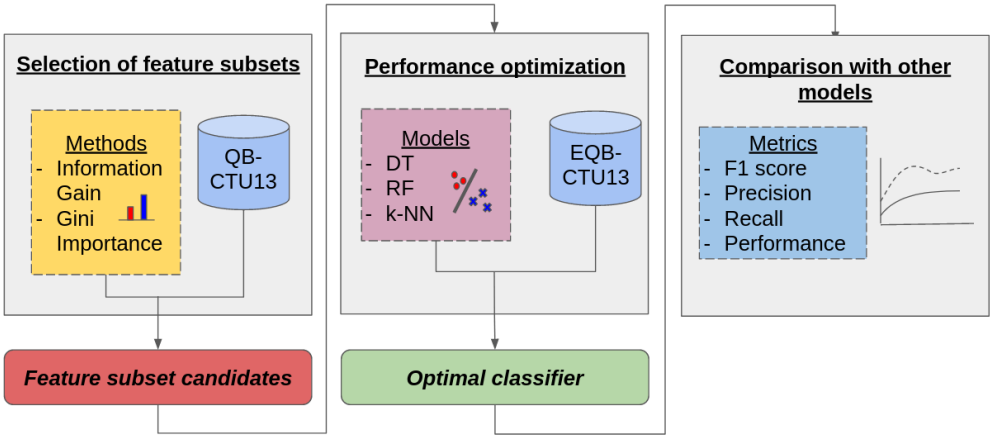

FIGURE 1. We used Information Gain and Gini Importance over the QB-CTU13 dataset to select three feature subsets from an initial set of eleven features. Next, we evaluated the three subsets along with three classifiers –DT, RF and k-NN– over the EQB-CTU13 dataset. Finally, we compared the most optimal proposal with two State-of-the-Art models.

modifications. The CTU-13 is a public dataset created in the Czech Technical University (CTU) [29], which contains data from seven different botnets: Neris, Rbot, Virut, Murlo, NSIS, Donbot and Sogou. Besides, CTU-13 can either be used *as a whole* or *by scenario*: CTU-13 contains 13 different scenarios, i.e., traffic captures carried out in different situations, where the monitored botnet performs one or more actions such as communication between bots, generating SPAM, or performing a DDoS attack.

The current purpose of the CTU-13 dataset is not to improve the detection rate of a previous work. In the recent literature, we found that the minimal accuracy obtained in all the scenarios of the CTU-13 dataset is 96.43%, achieved with a Self Organizing Map (SOM) [41]. Therefore, the current main utility of this dataset is to verify the detection ability of a solution.

Other botnet types can be used to test if a traffic classifier can improve a previous result. For example, we found that in the work of Abraham *et al.* [42] the best F1 score obtained for the botnet Bunitu was 90% with an ensemble of classifiers. Moreover, the work of AlAhmadi and Martinovic [43] reported that the recall obtained for the NotPetya botnet using an RF classifier was 60%, and that 25% of the samples of this botnet were wrongly classified as produced by the botnet Miuref.

B. FEATURE SELECTION

In general, feature selection techniques can be categorized as wrapper, filter, and embedded methods [44].

Wrapper methods search the most optimal feature subset for a given learning algorithm through iteration and evaluating the performance of feature subsets with that algorithm. Their main downside is that the feature subset they obtained is specifically selected for that learning algorithm, and it is not guaranteed to be working well for any different algorithm [45].

Filter methods overcome this problem by not evaluating the feature subset with a learning algorithm. Instead, they rely on characteristics of the data itself, such as entropy or feature correlations. As a result, these methods are usually more computationally efficient than wrapper methods, but the obtained feature subset may not be the most optimal for a given learning algorithm [46].

Finally, the embedded methods entail a mixture of the two previous ones. The performances of the feature subsets are evaluated with a pre-selected learning algorithm, but it does not use an iterative search as wrapper methods do. Instead, they assign weights to the initial features and try to minimize (or nullify) the value of the weights, which represent the feature importance [47].

In the context of feature selection for botnet detection, different methods have been applied. For example, the work of Gadelrab *et al.* [32] used the Weka framework to implement an embedded feature selector using "BestFirst" as search algorithm and "CfsSubsetEval" as evaluating method. This allowed the creation of feature subsets, which were afterwards evaluated with a DT classifier and three SVM classifiers (i.e., with linear, polynomial or Radial Basis Function (RBF) kernels). With an F1 score of 95%, the best result was achieved with a subset of five features and the DT classifier. The dataset used was built by the authors and was limited to IRC and HTTP based botnets.

Alauthaman *et al.* [48] tested three feature reduction methods: entropy impurity, RelieF and Principal Component Analysis (PCA). The target classification algorithm was a Neural Network (NN), and the most efficient result was obtained using the entropy impurity. The achieved accuracy was 99.20% after reducing the feature set from 29 to 10 features. The data used was a mixture of traffic traces from the ISOT [38] and the ISCX [49] datasets.

Moreover, Catak [50] compared five classifiers for malware traffic detection: DT, RF, NN, AdaBoost and NB. They

used a filter-based feature selection based on the L1-norm to reduce the feature pool from 36 to 11 features, achieving the highest accuracy with RF (89.04%) followed by DT (86.77%). The dataset used was created by the Australian Centre for Cyber Security (ACCS) [51].

The work of Letteri *et al.* [27] focused on improving the F1 score through feature reduction on a NN with two feature selection methods: a Gini Importance derived selection, and Information Gain (also known as Mutual Information). The features were based on time windows by counting the number and type of packets sent to a certain IP inside a time interval. The best results achieved an F1 score of 97.30% with both the Gini Importance and Information Gain based selections. The dataset used in this work was generated from traffic captures offered by the Stratosphere IPS [52].

Feng *et al.* [53] used PCA and an embedded method based on SVM to determine the importance of an initial set of 55 features for cyberattacks detection. The selected datasets were the C08, C09 and C13 ones from the CCC dataset collection [54], and the feature subsets were evaluated using SVM and RF. They determined the top 10 most important features and concluded that they could reduce the total set of 55 features to 40 without experiencing a significant decrease in the detection rate. After this work, Muhammad *et al.* [24] used the CCC dataset collection and also selected a subset of 40 features, but using PCA and Information Gain for feature selection. They tested more models, i.e., RF, SVM, Logistic Regression (LR) and MLP, and were able to overcome the result of Feng *et al.* using RF and to obtain the highest accuracy of 99%.

In addition, DL methods can also be used in feature reduction and selection. For example, the work of Parker *et al.* [55] used an AutoEncoder (AE) to obtain a set of 50 new features derived from the original set of 154 features. They combined the two sets and ranked the importance of the 204 features with the Information Gain method. To conclude, they determined that the most optimal solution was an LR classifier using five features, obtaining an F1 score of 98.01% with a computational training cost of 603.33 seconds on a computer with a 3.1 GHz Intel Core i7 CPU and 8 GB of RAM. The dataset they used was the reduced CLS portion of the AWID dataset [56].

Finally, state-of-the-art works show that it is possible to implement botnet detectors using a hyper-reduced set of features. Joshi *et al.* [57] used five feature selection techniques producing each a reduced feature subset from their original set. The smallest subset was obtained using PCA, and it had only two features: the source IP and the protocol. Besides, they tested the feature subsets with four classifiers, SVM, LR, k-NN and DT. Using the CTU-13 dataset, k-NN yielded the best results: 99% of accuracy using only the subset with two features. A year later, Ismail *et al.* [58] conducted a similar research using more recent data. They used seven traffic captures from the Stratosphere IPS [52], and they also tested more classifiers: Bayesian Network, NB, k-NN, Ada Boost, SVM, J48, RF, Ripper and PART. They concluded that SVM attained the best true positive rate on detecting botnet data across the seven traffic captures, using a subset of only three features: the destination and source IPs and the protocol. Despite these works providing insights on reduced feature sets, they did not analyze the computational cost of their proposals.

III. METHODOLOGY

For selecting a feature subset capable of maintaining a reasonable detection rate and reducing the time spent on the classification by the ML models, we elaborated the following methodology. First, as recommended by [20] we used a wrapper method to select the three features that contribute the most to the F1 score. One common wrapper strategy usually followed is to rank the importance of each feature, and then to check the evolution of the F1 score when adding the features one by one according to such ranking [22], [23], [28]. Sometimes, this strategy can be improved by ranking the features with two different methods, as it was done in [24] where they used both PCA and Information Gain to rank the features they selected to classify botnet traffic.

However, it is also known that using PCA on supervised learning can be counterproductive [59]. Because of that, in this work, we have used Gini Importance instead of PCA, since it measures how well can the data be divided into homogeneous groups, which is important for the three classifiers used in our experiments. Finally, after selecting the three subsets with the features that contribute the most to the F1 score, we measured the trade-off between the F1 score and the computational cost, in terms of the average time needed to classify a sample employing the trained models using each of these subsets.

A. FEATURE RANKING METHODS

Before selecting a reduced feature subset, it is necessary to determine the importance of each feature. As we just justified above, we used concurrently two methods to rank features: Gini Importance and Information Gain.

1) GINI IMPORTANCE

The Gini Gain is defined as the variation of the Gini Impurity after a split of the data using a feature [60]. The higher the decrease of the Gini Impurity, the more *important* the Gini Gain is perceived, and thus, we can rank the features according to this *Gini Importance*.

The Gini Impurity of a subset of samples Γ is defined in (1), where $p(i)$ is the probability of picking a subset sample of class i, and J is the number of classes present in the subset.

$$\Gamma = \sum_{i=1}^{J} p(i)(1 - p(i)) \tag{1}$$

Finally, the Gini Impurity of the whole dataset Γ_X in a given state, before or after one or more splits, is the weighted mean of the Gini Impurity of its subsets Γ_j as shown in (2),

TABLE 1. Network Traffic Features [14].

Feature(s)	Description
sPort, dPort	Source and destination ports in the TCP connection
mLen, vLen	Mean and variance of the payload lengths of all packets in the TCP connection
mTime, vTime	Mean and variance of the interval times between consecutive sent packets in the TCP connection
mResp, vResp	Mean and variance of the interval times between a packet reaches the machine that started the connection and it responds to that packet
nBytes	Total number of bytes exchanged in the TCP connection
nSYN	Total number of SYN flags exchanged during the TCP connection
nPackets	Total number of packets exchanged in the TCP connection

where w_j represents the j-th subset weight.

$$\Gamma_X = \frac{1}{n}\sum_{j=1}^{n} w_j \Gamma_j \tag{2}$$

2) INFORMATION GAIN

Information Gain allows us to quantify which feature provides maximal information about the classification, using for that the notion of entropy. The Information Gain obtained with a feature is defined as reducing the entropy in the dataset after knowing the values of the samples for this feature [61].

On the one hand, the original entropy of the dataset $H(X)$ is defined in (3), based on the probability $p(x)$ of a sample to belong to the class x. On the other hand, the entropy $H(X|Y)$ of the dataset after knowing the values of the feature Y is defined in (4) based on the probability $p(y)$ of a sample to present the value $y \in Y$, and the probability $p(x|y)$ of a sample of class x to present the feature value $y \in Y$.

$$H(X) = -\sum_{x=1}^{X} p(x) log(p(x)) \tag{3}$$

$$H(X|Y) = -\sum_{y} p(y) \sum_{x}^{X} p(x|y) log(p(x|y)) \tag{4}$$

B. FEATURE VECTOR EVALUATION

For this work, we only considered traffic based on the TCP protocol because it is used by the majority of known botnets [62]. In our previous work [14], we used thirteen features to characterize TCP flows, but in this work we have dismissed two of them: the average and the variance of the packets' velocity transmission (mVel and vVel). We removed them because their information was already represented by the average and variance of the interval time between consecutive packets (mTime and vTime), as shown in (5). Table 1 shows the remaining eleven features we selected to model network traffic.

$$\text{mVel} = \frac{\text{number of packets in TCP flow}}{\text{duration of TCP flow}} = \frac{1}{\text{mTime}} \tag{5}$$

We used three measures to evaluate different subsets of features: (1) the F1 score, to measure the capability to detect botnet samples, (2) the computational time, to consider the cost of the classification of a sample, and (3) the performance expressed as a relation between the former two.

The F1 score is defined as the harmonic mean of the precision and the recall [63]. In the context of this work, the precision is the fraction of samples detected as "botnet" that are actually generated by malware, while the recall is the fraction of botnet samples that are detected, over all the botnet samples in the dataset [64].

The computational time is the total time required by a computer with a specific processor to conclude a task. In this work, we used this time to measure the computational cost of classifying a sample, considering that all the experiments were made with the same CPU. The measured time did not include either the training time of the model or the time required to calculate the traffic features since they were already present in the datasets.

To measure the performance of a vector of n features, we use the ratio between the F1 score and the computational time, to allow the measurement of the gain in detection capability with respect to the computational cost of this detection.

C. CLASSIFIERS

In this work, we have carried out our experiments with three Machine Learning classifiers: Decision Tree (DT), Random Forest (RF) and k-Nearest Neighbors (k-NN).

The DT can be visualized as a tree-structured set of questions or decisions about the sample to be classified [65]. Depending on the values of the sample features, a path in the tree of decisions will be followed to estimate the class of the sample.

The RF classifier is a conglomerate of m DT classifiers [66]. In the training phase, each DT is trained with a bootstrap sample [67] of the training data. In the classification process, the output is generated with a voting system among all the DT models of the conglomerate.

The third algorithm we have assessed is k-NN. If the samples are characterized with m features, in the training phase, the k-NN classifier stores the samples in a m dimensional hyperspace. In the classification phase, each sample is projected in that hyperspace and the algorithm searches for the k nearest training samples to determine its class [68].

IV. EXPERIMENTATION

A. DATASETS

The experimentation was made using two datasets *QB-CTU13* and *EQB-CTU13*, which we built based on the CTU-13 dataset [29] to overcome its class imbalance. The significant differences in the number of usable TCP flows of CTU-13 may suppose an issue in the multi-classification since the algorithms tend to over-fit the most represented classes. The least represented botnet class (Sogou) has only 36 usable TCP flows, so balancing the dataset to this class would mean a significant waste of data. Not considering the

TABLE 2. TCP flows used as Class Samples in QB-CTU13 and EQB-CTU13 Datasets.

Class	QB-CTU13	EQB-CTU13
Normal	3890	3890
Neris	3890	3890
Rbot	3890	3890
Virut	3890	3890
Murlo	2036	2036
NSIS	355	355
Donbot	233	233
Sogou	36	36
Bunitu	-	3890
Miuref	-	3890
NotPetya	-	111

smaller classes would reduce the dataset to hold only data from three botnets.

The *Quasi-Balanced based on CTU-13 dataset* (QB-CTU13) contains the same number of usable TCP flows from the three most represented botnets, and all the usable TCP flows from the remaining ones, hence the name 'Quasi-Balanced'. We developed this dataset because it allowed us to obtain a more meaningful weighted F1 score since the results of the most represented classes do not "hide" the ones of the less represented ones.

The *Extended QB-CTU13* (EQB-CTU13) is constructed using the same TCP flows from the QB-CTU13, and TCP flows from the Stratosphere IPS [52] of three new botnet classes: Bunitu, Miuref and NotPetya. We used all the available samples from these three botnets in their repository by the end of 2020. We selected these new particular botnets to be able to compare our study with the works of Abraham *et al.* [42] and AlAhmadi and Martinovic [43]. The number of used TCP flows of each class in both datasets is specified in Table 2. We obtained a sample from each flow as pictured in Fig. 2, where we also show the features we extracted.

B. FEATURE EVALUATION AND FEATURE SUBSET SELECTION

The experimentation carried out in this work was accomplished on a machine with 128 GB of RAM and two Intel(R) Xeon(R) E5-2630v3 CPUs running an Ubuntu 14.04.5 LTS. To run the classifiers, we developed our own scripts using Python 3 and the library scikit-learn.[1]

As we already explained, to evaluate the information provided by each of the eleven features described in Subsection III-B, we used two filter selection techniques, i.e., Gini Importance (GI) and Information Gain (IG), to rank them over the QB-CTU13 dataset from these two different perspectives. Next, we analyzed the evolution of the F1 score over the same dataset when introducing a new feature, i.e., increasing the number n of used features, from $n = 1$ to $n = 11$.

[1] http://scikit-learn.org/stable/index.html

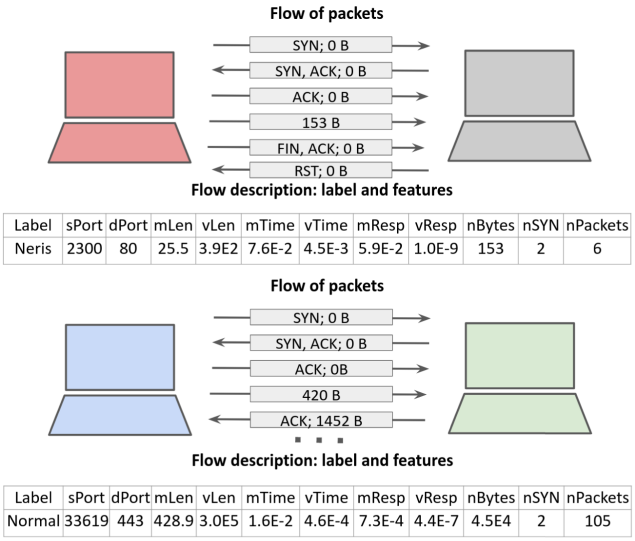

FIGURE 2. When two computers establish a communication using the TCP protocol, they interchange a flow of packets. The features extracted from a flow constitute a sample. On the top case, an infected computer connects to a server, and on the bottom the connection is made between normal computers.

We experimented with the two different ways of adding features, i.e., following the GI and IG rankings, and we obtained the F1-score with DT, RF and k-NN. In each of these tests, the F1 score was calculated as the weighted average of the results of a 10-fold cross-validation.

The DT was optimized by not limiting the number of generated branches. With respect to RF and k-NN, we searched for the most optimal versions by using a different number, m, of DTs ($m \in \{10, 20, \ldots, 100\}$) and different amount, k, of neighbors ($k \in \{1, 3, 5, 7, 9\}$), respectively.

By observing the evolution of the F1 score, we built three feature subsets containing the n, $n-1$ and $n-2$ features that produce the most notable increases in the F1 score among the three classifiers. Finally, we used the EQB-CTU13 dataset to study the performance of the three optimized classifiers using each one of the three feature subsets. Since the F1 score is calculated as the average results in a 10-fold cross-validation, we keep the same data proportion to calculate the computational time. Then, the mean time was calculated using 10% of the samples and the model was trained with the other 90% of the dataset.

V. RESULTS AND DISCUSSION

As we pointed out before, we analyzed the importance of the eleven selected features with two feature selection methods, Gini Importance (GI) and Information Gain (IG), over the QB-CTU13 dataset, as shown in Fig. 3. We observed the disparity between the source (sPort) and destination (dPort) ports. We assumed this fact was reasonable since the dPort is related to the type of service used in the connection, e.g., port 80 is usually linked to HTTP traffic. However, the sPort can be changed by a Network Access Translation (NAT) system or auto-assigned by the Operating System without the

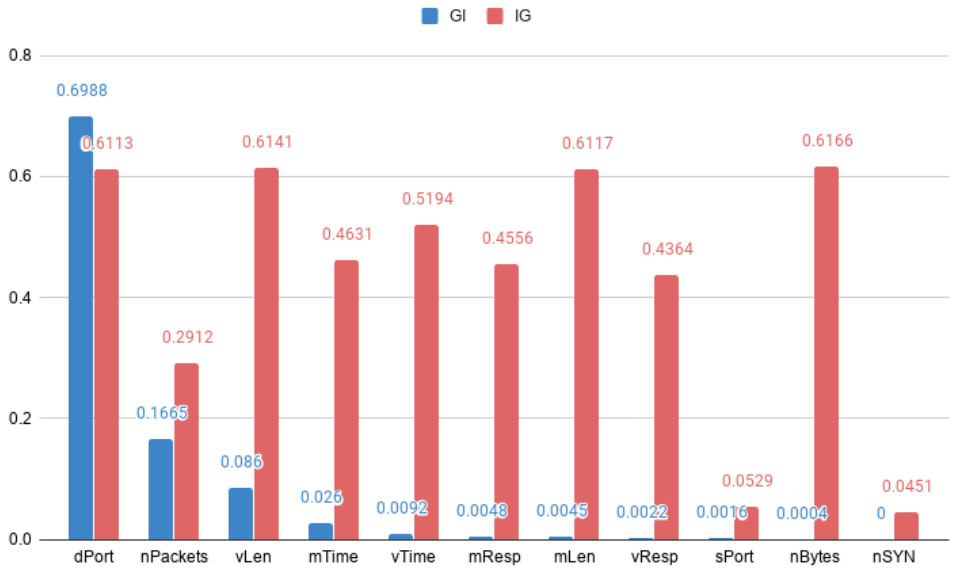

FIGURE 3. Gini Importance (GI) and Information Gain (IG) of the 11 selected features over the QB-CTU13 dataset.

intervention of the program that originated the connection. Besides, we observed that both selection methods consider the number of exchanged SYN flags (nSYN) as the least important feature.

Once we obtained the two feature rankings shown in Fig. 3, we measured the evolution of the weighted F1 score when an increasing number of features is considered. We added a feature each time, one by one, from the most important (GI), see Fig. 5, or most informative (IG), see Fig. 6, to the least relevant.

The F1 scores were calculated using DT, RF and k-NN, looking for the most optimal versions in terms of performance of the RF and k-NN. In the case of RF, we experimented with different numbers of DT ($m \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$), and in the case of k-NN, we considered several numbers of neighbors ($k \in \{1, 3, 5, 7, 9\}$). Fig. 4 shows the F1 scores obtained with each version of the RF and the k-NN. It is observable that there was little effect by varying the values m for RF and k for k-NN. Thus, we selected the smaller values since they favored the speed of the classifiers. To compare RF and k-NN with DT, in Fig. 5 and Fig. 6 we only represented the results of the optimal values $m = 10$ for RF and $k = 1$ for k-NN, and also because it improved the data visualization.

Fig. 5 shows the gradual increase of the F1 score by adding features following the order dictated by the GI ranking. We observed that dPort, nPackets, vLen, mTime and sPort are, in this order, the features that generated the most significant increment of the F1 score.

Similarly, when observing in Fig. 6 the growth of F1-score by adding features following the IG order, the features that caused the most significant F1 raises were nBytes, vLen, mLen, dPort, vTime and sPort.

As explained previously, the sPort might be hidden behind a NAT service, and therefore we did not consider this feature to be a candidate for the proposed optimal subsets. Therefore, as candidate features, we selected the ones that, by being added, produced the most significant increases on the F1 score in the previous experiments: dPort, vLen, nBytes, nPackets, mLen, mTime and vTime, making a total of $n = 7$ features.

We selected three slightly different subsets of features to search for the optimal performance containing five, six and seven features each. The initial five-features subset is composed of the two most relevant features for each method of selection used, that in the case of GI are dPort and nPackets and for IG are nBytes and vLen. We also included mLen, which is the third one proposed by IG, because its information coefficient is very close to vLen, the second one. The fourth one in the IG list is dPort, already included since it is the first one in the GI list. We decided to exclude the fifth one, vTime, because its coefficient was not so close to the previous four features and its relevance, in the GI list, is very small.

We decided to include mTime and vTime for the seven features subset, because they are the two following most relevant features in both lists. Moreover, to evaluate the performance with a smaller feature vector, we decided to drop vTime for the six-features vector since its GI coefficient is very low.

Therefore, the three different feature vectors evaluated were the following ones:

1) Vector with five features: [dPort, nPackets, nBytes, vLen, mLen]

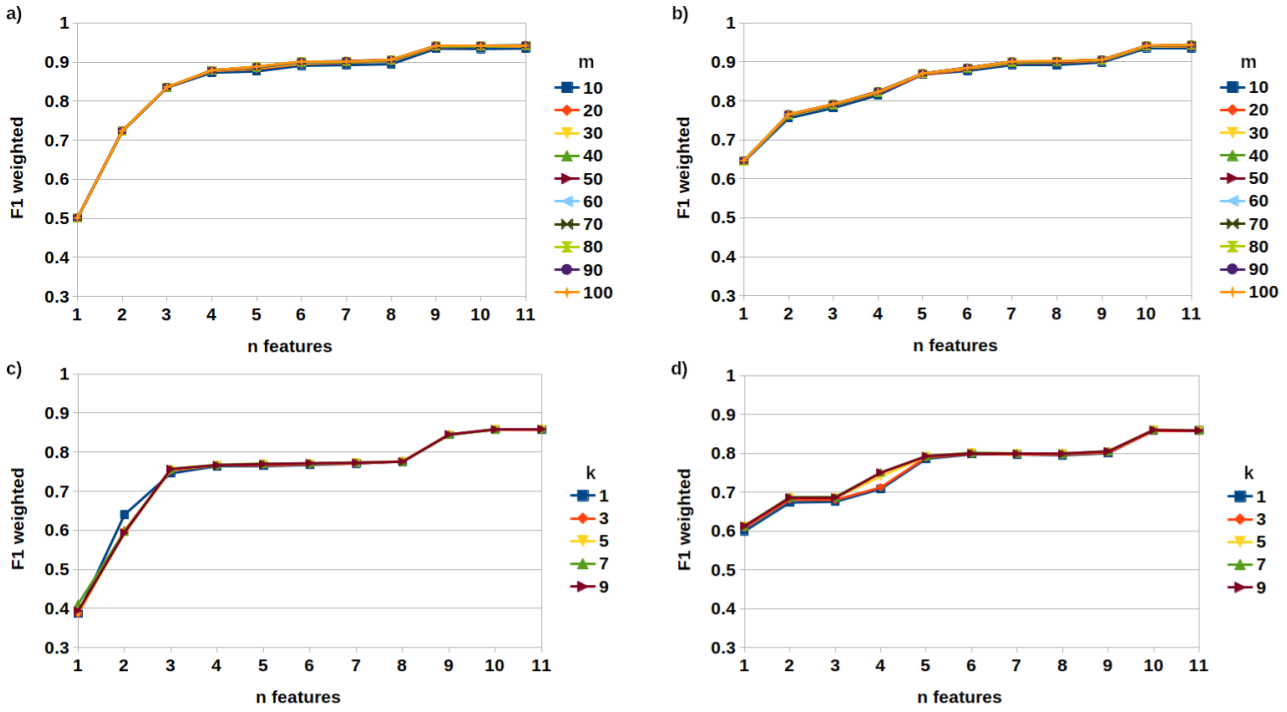

FIGURE 4. Evolution of the F1 score by adding features to classify the traffic. On the top graphics (a and b) we compare the F1 score of different versions of RF integrating *m* DTs, and on the bottom (c and d) we compare different versions of k-NN by varying the number *k* of neighbors. On the left-hand side graphics (a and c), the features were added following the Gini Importance, while on the right ones (b and d) they were added following the Information Gain. All the scores were obtained using a 10-fold cross-validation.

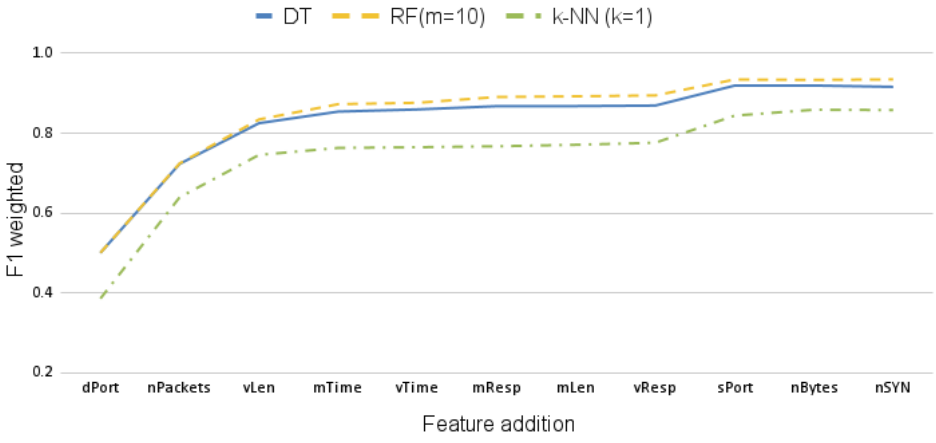

FIGURE 5. Evolution of the F1 scores by increasing the number of considered features following the Gini Importance ranking. The *x* axis shows the feature that is added: First, the models started only using dPort, then they used dPort and nPackets, and so on. In the case of RF and k-NN, only the optimal values of *m* and *k* are represented.

2) Vector with six features: [dPort, nPackets, nBytes, vLen, mLen, mTime]
3) Vector with seven features: [dPort, nPackets, nBytes, vLen, mLen, mTime, vTime]

These selected feature subsets were tested with the three optimized models, DT, RF ($m = 10$) and k-NN ($k = 1$) over the EQB-CTU13 dataset. Fig. 7 represents the weighted F1 scores, and shows a clear advantage of DT and RF over k-NN. Moreover, k-NN also showed a higher computational cost for classification than the other models (see Fig. 8). Besides, it is expected that it will perform even worse in computational time with larger datasets, since it searches for the k most similar samples among all the training data. On the other hand, while RF achieved higher F1 scores than DT

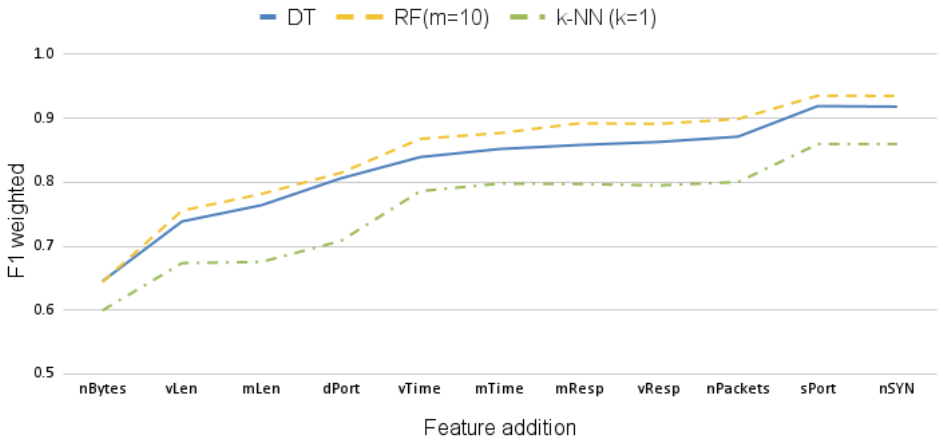

FIGURE 6. Evolution of the F1 scores by increasing the number of considered features following the Information Gain ranking. The *x* axis shows the feature that is added: First, the models started only using nBytes, then they used nBytes and vLen, and so on. In the case of RF and k-NN, only the optimal values of m and k are represented.

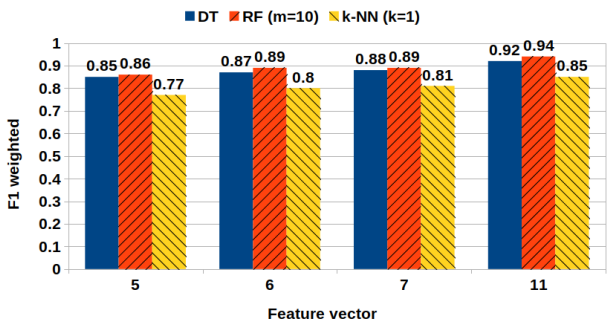

FIGURE 7. F1 scores of DT, RF ($m = 10$) and k-NN ($k = 1$) using the 5, 6 and 7-features subsets, as well as using the complete set of eleven features. The F1 used a weighted average among classes of the EQB-CTU13 dataset.

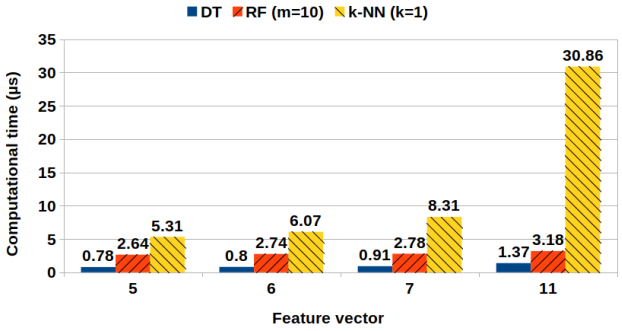

FIGURE 8. Computational time to classify a sample needed by DT, RF ($m = 10$) and k-NN ($k = 1$) using the 5, 6 and 7-features subsets and the complete set of eleven features over the EQB-CTU13 dataset.

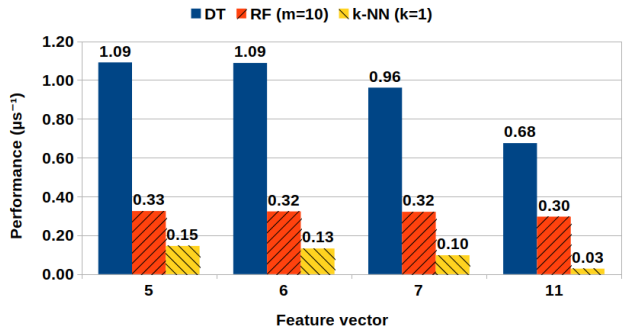

FIGURE 9. Performance achieved by DT, RF ($m = 10$) and k-NN ($k = 1$) using the 5, 6 and 7-features subsets and the complete set of eleven features over the EQB-CTU13 dataset.

(see Fig. 7), the cost of using 10 DTs instead of one meant a significant reduction in the performance, as shown in Fig. 9. The tested model that achieved the best performance was DT using the 5-feature subset, and the DT using a 6-feature subset achieved a close result. However, using an extra feature also means that it has to be calculated. Thus, under the perspective of performance, the DT with the 5-feature subset is more recommendable.

On Table 3 we show the F1, recall, precision and performance values by each class of the EQB-CTU13 dataset achieved in a 10-fold cross-validation with two versions of the DT: using the performance-optimal 5-features subset, and using all the eleven features. The performance was defined as the ratio between the F1 score and the time required to classify a sample. We also compared our proposal with the state-of-the-art proposals of Joshi *et al.* [57], i.e., a k-NN using two features (knn-2f); and Ismail *et al.* [58], i.e., an SVM using three features (svm-3f). According to the experimentation of these two works, k-NN and SVM obtained better results than DT using their proposed two and three feature sets respectively.

Regarding the two approaches based on DT (i.e., dt-5fs and dt-11fs), in three botnet classes, i.e., Donbot, Murlo and NotPetya, the F1 score achieved with five features is better than that obtained using the eleven features. The recall score obtained for NotPetya is higher when using the 5-feature subset. Although the eleven feature vector achieves better detection results, the performance metric is significantly higher

TABLE 3. F1, Recall, Precision and Performance scores of our proposal, a DT with a 5-feature subset (dt-5fs). For comparison, we include the scores of a DT using the initial set of 11 features (dt-11fs), the SVM proposed by Joshi *et al.* [57] which uses three features (svm-3fs), and the k-NN proposal of Ismail *et al.* [58] which uses two features (knn-2fs).

	F1				Recall				Precision				Perf. ($m.s^{-1}$)			
Class	dt-5fs	dt-11fs	svm-3f	knn-2f	dt-5fs	dt-11fs	svm-3f	knn-2f	dt-5fs	dt-11fs	svm-3f	knn-2f	dt-5fs	dt-11fs	svm-3f	knn-2f
Normal	0.78	0.93	0.99	0.99	0.83	0.93	0.99	0.99	0.74	0.93	1.00	1.00	**1007.49**	68.32	2.16	48.51
Bunitu	0.84	0.94	0.01	0.99	0.87	0.94	0.07	0.99	0.81	0.94	0.01	0.99	**1079.65**	682.32	0.03	48.46
Donbot	0.88	0.85	0.64	0.98	0.85	0.87	0.52	0.98	0.91	0.83	0.84	0.99	**1139.20**	620.69	1.38	47.97
Miuref	0.99	1.00	0.37	1.00	0.99	1.00	0.23	1.00	0.99	1.00	1.00	1.00	**1278.51**	728.21	0.81	48.76
Murlo	0.98	0.97	0.00	1.00	0.97	0.97	0.00	1.00	0.99	0.97	0.00	1.00	**1259.58**	709.21	0.00	48.70
NSIS	0.59	0.85	0.00	0.01	0.59	0.86	0.00	0.11	0.59	0.84	0.00	0.10	**761.88**	623.66	0.00	0.29
Neris	0.69	0.84	0.00	1.00	0.71	0.83	0.00	1.00	0.67	0.85	0.00	1.00	**891.02**	610.92	0.00	48.63
NotPet.	0.96	0.95	0.00	0.99	0.97	0.95	0.00	1.00	0.95	0.94	0.00	0.98	**1241.69**	693.49	0.00	48.28
Rbot	0.98	0.99	0.50	0.96	0.98	0.99	0.58	0.92	0.98	0.99	0.45	0.99	**1262.78**	722.94	1.09	46.62
Sogou	0.22	0.55	0.00	1.00	0.38	0.48	0.00	1.00	0.15	0.64	0.00	1.00	285.76	**400.61**	0.00	48.78
Virut	0.75	0.84	0.32	1.00	0.78	0.85	0.56	1.00	0.72	0.83	0.22	1.00	**967.83**	614.89	0.69	47.71

with the five features vector, except for the class Sogou, which is explainable by the scarce representation of this class (i.e., it only contains 36 TCP flows). This also caused the botnet Sogou to obtain the lowest values of F1 score and recall. Other works [41] obtained a higher score for this botnet in the CTU-13 dataset since they worked with each scenario separately.

As we indicated before, to create the EQB-CTU13 dataset from the QB-CTU13, we added traffic samples from the Bunitu, NotPetya and Miuref botnets in order to compare our models with previous results from other authors. In Abraham *et al.* [42], the best F1 score achieved for the Bunitu botnet was 90%, while with our approach, using eleven features, we obtain an F1 of 94% and, with the most time-efficient subset, with five features, the F1 score achieved was 84%.

Concerning NotPetya and Miuref botnets, AlAhmadi & Martinovic [43] obtained a recall for the NotPetya botnet of 60%, where 25% of the NotPetya samples were wrongly classified as Miuref samples. In our proposal, the performance-optimized DT (i.e. dt-5fs) achieved a recall of 97% for the NotPetya class, and the F1 and recall scores of the Miuref class were both of 99%.

Finally, on the one side, the k-NN using two features attained the best results in all the classes. These two features were the source IP and the protocol used. On the other side, the SVM also used these two features and added the destination IP, and yielded worse results. It is observable that the SVM using three features attains a good performance distinguishing between normal and botnet traffic, but it suffers in the task of differentiating botnets. However, the performance of these two classifiers was in the order of 10^{-2} lower than the performance of the DT, which indicated that even using more features and also considering the extra time to calculate them, the DT is significantly faster than SVM and k-NN.

VI. CONCLUSION AND FUTURE WORKS

In this work, we propose a reduced set of features to detect and classify, with high accuracy, some of the most popular bot networks. Our proposal is more efficient in terms of time and obtains a similar or better accuracy than other approaches provided for classifying these botnets.

We selected the most relevant features using the Gini Importance and Information Gain criteria. With the selected features, we proposed three subsets, evaluating its performance using Decision Trees (DT), Random Forest (RF) and K-Nearest Neighbors (k-NN) for botnet detection. We optimized RF and k-NN tuning their respective parameters m – i.e., the number of employed DTs – and k – i.e., the number of nearest neighbors to consider.

In order to carry out the experimentation, we crafted two datasets, QB-CTU13 and EQB-CTU13. Both were based on the CTU-13 dataset, and they were designed to overcome its limitations regarding the imbalance of its classes. Moreover, in EQB-CTU13, we added three new botnet classes, Bunitu, Miuref and NotPetya, to compare our most optimal model with other works.

Initially, we selected eleven of the most used features, we ranked their relevance according to the Gini Importance and Information Gain. Then, we obtained a list of the most informative features from the point of view of each method based on the information present in the QB-CTU13 dataset.

Using this dataset, we analyzed the progression of the F1 score by adding features in the order established by each of the two rankings. We observed that seven features produced the biggest increase in the F1 score. However, we still evaluated three subsets of five, six and seven features to determine which one offered the best balance between computational time and F1 score. Regarding the evolution of the F1 score, we noticed that changes in the parameters m and k did not make a significant difference. Therefore, we selected $m = 10$ and $k = 1$ because they were the least computationally expensive.

After selecting the three feature subsets and the optimal parameters for the models, we tested them with the QB-CTU13 dataset. The evaluation was focused on the performance, stated as the ratio between the weighted F1 score and the computational time required to classify a sample. According to the results, DT with the 5-feature subset reached the best trade-off between detection ability and computational cost, surpassing the other two models and the other two feature subsets.

We compared the selected model, i.e., DT with the 5-feature subset, with another DT using all the eleven features. For this comparison, we employed the EQB-CTU13

dataset because it includes three more botnet classes, Bunitu, NotPetya and Miuref, allowing us to compare our models with other recent works. We found that the 5-feature subset achieved better F1 results than the whole feature set in three classes: Donbot, Murlo and NotPetya. Additionally, the recall of the 5-features DT on the Miuref and NotPetya classes – i.e., 99% and 97% – was significantly higher than the recall reported in the literature. The 90% of F1 score for the Bunitu class reported in the found literature was not surpassed by the DT with the optimized feature subset, with only an F1 of 84%. However, it was overcome by our model with eleven features using DT, with an F1 score of 94%.

Based on these results, we consider that our proposal using DT with the five selected features is the best solution for a multi-class botnet detector focused on performance. Using a single DT may indeed produce over-fitting, and in some situations, it could be better to sacrifice time performance and use a Random Forest. However, in our proposal, we avoided over-fitting using a small subset of features and testing the model with a cross-validation scheme.

We also compared our proposal with two state-of-the-art models: the SVM proposed by Joshi *et al.* [57] and the k-NN proposed by Ismail *et al.* [58]. In contrast with our proposal, both models rely on the protocol and the IPs and as features. We did not consider the protocol since we already used the ports, which are usually used to identify the protocol, especially if the connection is ciphered with SSL/TLS or similar. We also did not consider the IP as a feature for three reasons: first, the IPs are usually assigned to the device by a DHCP service or are pre-configured, and thus the malware cannot change them and so the IP should not give information about the malware. Second, the captured IPs might be masked by a NAT service or a proxy service. And third, we consider that, since a given IP identifies a computer, then a model trained with this kind of feature could be over-fitted to the used dataset. For example, if in the dataset we have a computer with IP 10.0.0.5 and it is infected with Miuref, then the model might learn that ''any computer with IP 10.0.0.5 is infected with Miuref''. This statement is false, since a computer assigned with the IP 10.0.0.5 can be clean of viruses in a different intranet. As future work, we plan to conduct an experiment to provide further evidence of this situation.

Finally, we want to remark that the obtained results show that k-NN is not suitable for this problem since its computational cost is quite high. Even using just two features, the model was about 100 times slower than DT using five features. Moreover, it increases considerably with the number of samples used in the training phase.

We consider that a study on botnet detection over high bandwidth traffic could be interesting in future works. However, in this scenario, the large number of packets sent per second makes it near impossible to analyze them all in real-time without dropping packets. Therefore, it would be necessary to determine the minimal requirements to detect botnet traffic in the most challenging conditions. Additionally, we consider that the model performance can be further improved by using a compiled language such as C/C++ or Rust.

REFERENCES

[1] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, ''Botnet detection based on traffic behavior analysis and flow intervals,'' *Comput. & Secur.*, vol. 39, pp. 2–16, 2013.

[2] L. Wang, L. Pepin, Y. Li, F. Miao, A. Herzberg, and P. Zhang, ''Securing power distribution grid against power botnet attacks,'' in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Aug. 2019, pp. 1–5.

[3] K. Bissell, R. M. Lasalle, and P. Dal-Cin. (Mar. 2019). *Ninth Annual Cost of Cybercrime Study*. Accenture. [Online]. Available: https://www.accenture.com/us-en/insights/security/cost-cybercrime-study

[4] FBI's Internet Crime Complaint Center (IC3). (Feb. 2020). *2019 Internet Crime Report*. IC3. [Online]. Available: https://www.fbi.gov/news/stories/2019-internet-crime-report-released-02%1120

[5] M. B. Lourenço and L. Marinos, ''ENISA threat landscape 2020—Botnet,'' ENISA, Attiki, Greece, Tech. Rep., Oct. 2020, no. 26.

[6] N. Koroniotis, N. Moustafa, and E. Sitnikova, ''Forensics and deep learning mechanisms for botnets in Internet of Things: A survey of challenges and solutions,'' *IEEE Access*, vol. 7, pp. 61764–61785, 2019.

[7] W. Li, J. Jin, and J.-H. Lee, ''Analysis of botnet domain names for IoT cybersecurity,'' *IEEE Access*, vol. 7, pp. 94658–94665, 2019.

[8] M. Roesch, ''Snort: Lightweight intrusion detection for networks,'' *Lisa*, vol. 99, pp. 229–238, Jun. 1999.

[9] E. Albin and N. C. Rowe, ''A realistic experimental comparison of the Suricata and Snort intrusion-detection systems,'' in *Proc. 26th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Mar. 2012, pp. 122–127.

[10] M. A. Aydın, A. H. Zaim, and K. G. Ceylan, ''A hybrid intrusion detection system design for computer network security,'' *Comput. Elect. Eng.*, vol. 35, no. 3, pp. 517–526, 2009.

[11] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, ''Intrusion detection system: A comprehensive review,'' *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.

[12] B. Anderson and D. McGrew, ''Identifying encrypted malware traffic with contextual flow data,'' in *Proc. ACM Workshop Artif. Intell. Secur.*, Oct. 2016, pp. 35–46.

[13] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, ''Anomaly-based network intrusion detection: Techniques, systems and challenges,'' *Comput. Secur.*, vol. 28, nos. 1–2, pp. 18–28, 2009.

[14] J. Velasco-Mata, E. Fidalgo, V. González-Castro, E. Alegre, and P. Blanco-Medina, ''Botnet detection on TCP traffic using supervised machine learning,'' in *Proc. Int. Conf. Hybrid Artif. Intell. Syst.* Berlin, Germany: Springer, 2019, pp. 444–455.

[15] M. Stevanovic and J. M. Pedersen, ''An efficient flow-based botnet detection using supervised machine learning,'' in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 797–801.

[16] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, N. A. Golilarz, and M. Alazab, ''An adaptive multi-layer botnet detection technique using machine learning classifiers,'' *Appl. Sci.*, vol. 9, no. 11, p. 2375, Jun. 2019.

[17] S. Lee, A. Abdullah, N. Jhanjhi, and S. Kok, ''Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning,'' *PeerJ Comput. Sci.*, vol. 7, Jan. 2021, Art. no. e350.

[18] M. Alshamkhany, W. Alshamkhany, M. Mansour, M. Khan, S. Dhou, and F. Aloul, ''Botnet attack detection using machine learning,'' in *Proc. 14th Int. Conf. Innov. Inf. Technol. (IIT)*, Nov. 2020, pp. 203–208.

[19] J. M. H. Jimenez and K. Goseva-Popstojanova, ''The effect on network flows-based features and training set size on malware detection,'' in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–9.

[20] A. Guerra-Manzanares, H. Bahsi, and S. Nomm, ''Hybrid feature selection models for machine learning based botnet detection in IoT networks,'' in *Proc. Int. Conf. Cyberworlds (CW)*, Oct. 2019, pp. 324–327.

[21] A. Guerra-Manzanares, S. Nomm, and H. Bahsi, ''Towards the integration of a post-hoc interpretation step into the machine learning workflow for IoT botnet detection,'' in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2019, pp. 1162–1169.

[22] L. Silva, L. Utimura, K. Costa, M. Silva, and S. Prado, ''Study on machine learning techniques for botnet detection,'' *IEEE Latin Amer. Trans.*, vol. 18, no. 5, pp. 881–888, May 2020.

[23] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Oct. 2014, pp. 247–255.
[24] A. Muhammad, M. Asad, and A. R. Javed, "Robust early stage botnet detection using machine learning," in *Proc. Int. Conf. Cyber Warfare Secur. (ICCWS)*, Oct. 2020, pp. 1–6.
[25] M. S. Koli and M. K. Chavan, "An advanced method for detection of botnet traffic using intrusion detection system," in *Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Mar. 2017, pp. 481–485.
[26] F. Beer and U. Buhler, "Feature selection for flow-based intrusion detection using rough set theory," in *Proc. IEEE 14th Int. Conf. Netw., Sens. Control (ICNSC)*, May 2017, pp. 617–624.
[27] I. Letteri, G. Della Penna, and P. Caianiello, "Feature selection strategies for HTTP botnet traffic detection," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Jun. 2019, pp. 202–210.
[28] F. V. Alejandre, N. C. Cortes, and E. A. Anaya, "Feature selection to detect botnets using machine learning algorithms," in *Proc. Int. Conf. Electron., Commun. Comput. (CONIELECOMP)*, Feb. 2017, pp. 1–7.
[29] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.
[30] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," in *Proc. 17th Conf. Secur. Symp.*, 2008, pp. 139–154.
[31] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proc. 16th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, Feb. 2008, pp. 235–252.
[32] M. S. Gadelrab, M. ElSheikh, M. A. Ghoneim, and M. Rashwan, "BotCap: Machine learning approach for botnet detection based on statistical features," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 3, pp. 563–579, 2018.
[33] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3.24, pp. 479–482, 2018.
[34] R. McKay, B. Pendleton, J. Britt, and B. Nakhavanit, "Machine learning algorithms on botnet traffic: Ensemble and simple algorithms," in *Proc. 3rd Int. Conf. Compute Data Anal.*, Mar. 2019, pp. 31–35.
[35] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of Twitter accounts: Are you a human, bot, or cyborg?" *IEEE Trans. Depend. Sec. Comput.*, vol. 9, no. 6, pp. 811–824, Nov. 2012.
[36] J. T. M. Garre, M. Gil Pérez, and A. Ruiz-Martínez, "A novel machine learning-based approach for the detection of SSH botnet infection," *Future Gener. Comput. Syst.*, vol. 115, pp. 387–396, Feb. 2021.
[37] S. Maeda, A. Kanai, S. Tanimoto, T. Hatashima, and K. Ohkubo, "A botnet detection method on SDN using deep learning," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–6.
[38] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. 9th Annu. Int. Conf. Privacy, Secur. Trust*, Jul. 2011, pp. 174–180.
[39] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! A case study on Android malware detection," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 4, pp. 711–724, Jul. 2019.
[40] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *Proc. Int. Conf. Artif. Intell. Big Data (ICAIBD)*, May 2018, pp. 228–234.
[41] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, "Unsupervised monitoring of network and service behaviour using self organizing maps," *J. Cyber Secur. Mobility*, vol. 8, pp. 15–52, Aug. 2018.
[42] B. Abraham, A. Mandya, R. Bapat, F. Alali, D. E. Brown, and M. Veeraraghavan, "A comparison of machine learning approaches to detect botnet traffic," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.
[43] B. A. Alahmadi and I. Martinovic, "MalClassifier: Malware family classification using network flow sequence behaviour," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*, May 2018, pp. 1–13.
[44] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, p. 94, 2016.
[45] A. Wang, N. An, G. Chen, L. Li, and G. Alterovitz, "Accelerating wrapper-based feature selection with K-nearest-neighbor," *Knowl.-Based Syst.*, vol. 83, pp. 81–91, Jul. 2015.
[46] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016.
[47] A. Jovic, K. Brkic, and N. Bogunovic, "A review of feature selection methods with applications," in *Proc. 38th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2015, pp. 1200–1205.
[48] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem, and M. A. Hossain, "A P2P botnet detection scheme based on decision tree and adaptive multilayer neural networks," *Neural Comput. Appl.*, vol. 29, no. 11, pp. 991–1004, 2018.
[49] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.
[50] F. O. Catak, "Two-layer malicious network flow detection system with sparse linear model based feature selection," *J. Nat. Sci. Found. Sri Lanka*, vol. 46, no. 4, p. 601, Dec. 2018.
[51] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[52] *Stratosphere IPS Website*. Accessed: Oct. 2019. [Online]. Available: https://www.stratosphereips.org/
[53] Y. Feng, H. Akiyama, L. Lu, and K. Sakurai, "Feature selection for machine learning-based early detection of distributed cyber attacks," in *Proc. IEEE 16th Int. Conf. Dependable, Autonomic Secure Comput., 16th Int. Conf. Pervas. Intell. Comput., 4th Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2018, pp. 173–180.
[54] O. Takata. (Oct. 2019). *MWS Datasets 2016 Anti-Malware Engineering Workshop (MWS)*. [Online]. Available: www.iwsec.org/mws/2016/20160714-takata-dataset.pdf
[55] L. R. Parker, P. D. Yoo, T. A. Asyhari, L. Chermak, Y. Jhi, and K. Taha, "DEMISe: Interpretable deep extraction and mutual information selection techniques for IoT intrusion detection," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–10.
[56] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st. Quart., 2016.
[57] C. Joshi, V. Bharti, and R. K. Ranjan, "Analysis of feature selection methods for p2p botnet detection," in *Proc. Int. Conf. Adv. Comput. Data Sci.* Berlin, Germany: Springer, 2020, pp. 272–282.
[58] Z. Ismail, A. Jantan, M. N. Yusoff, and M. U. Kiru, "The effects of feature selection on the classification of encrypted botnet," *J. Comput. Virol. Hacking Techn.*, vol. 17, no. 1, pp. 61–74, Mar. 2021.
[59] A. J. Das and S. Yaswanth. (2019). *Risks and Caution on Applying PCA for Supervised Learning Problems*. Accessed: Feb. 24, 2021. [Online]. Available: https://towardsdatascience.com/risks-and-caution-on-applying-pca-for-su%pervised-learning-problems-d7fac7820ec3
[60] S. Nembrini, I. R. König, and M. N. Wright, "The revival of the Gini importance?" *Bioinformatics*, vol. 34, no. 21, pp. 3711–3718, Nov. 2018.
[61] P.-P. Wen, S.-P. Shi, H.-D. Xu, L.-N. Wang, and J.-D. Qiu, "Accurate *in silico* prediction of species-specific methylation sites based on information gain feature optimization," *Bioinformatics*, vol. 32, no. 20, pp. 3107–3115, Oct. 2016.
[62] P. Wainwright and H. Kettani, "An analysis of botnet models," in *Proc. 3rd Int. Conf. Compute Data Anal.*, Mar. 2019, pp. 116–121.
[63] Y. Sasaki, "The truth of the F-measure," Univ. Manchester, Manchester, U.K., Lecture Notes, Oct. 2007. [Online]. Available: https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf
[64] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *Proc. AAAI Workshop Learn. Text Categorization*, vol. 752, no. 1, 1998, pp. 41–48.
[65] W. Buntine and T. Niblett, "A further comparison of splitting rules for decision-tree induction," *Mach. Learn.*, vol. 8, no. 1, pp. 75–85, Jan. 1992.
[66] A. Liaw and M. Wiener, "Classification and regression by random forest," *R Newslett.*, vol. 2, no. 3, pp. 18–22, 2002.
[67] K. Singh and M. Xie, *Bootstrap: A Statistical Method*. New Brunswick, NJ, USA: Manuscript, Rutgers Univ., 2008. [Online]. Available: http://www.stat.rutgers.edu/home/mxie/RCPapers/bootstrap.pdf
[68] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proc. 20th Int. Conf. World Wide Web (WWW)*, 2011, pp. 577–586.

JAVIER VELASCO-MATA received the B.Sc. degree in physics from the University of Granada, Spain, in 2016, and the M.Sc. degree in cyber-security from the University of León, in 2019. He is currently a Predoctoral Fellow with the Department of Electrical, Systems, and Automation, University of León. His current research interests include machine and deep learning applied to cyber-security.

VÍCTOR GONZÁLEZ-CASTRO received the B.S. and Ph.D. degrees in computer science from the University of León, Spain, in 2006 and 2011, respectively. He is currently an Associate Professor with the Department of Electrical, Systems, and Automation Engineering, University of León. His current research interests include computer vision, machine learning, and deep learning, applied to medical images, quality assessment, and cyber-security.

EDUARDO FIDALGO FERNÁNDEZ received the M.Sc. degree in industrial engineering and the Ph.D. degree from the University of León, in 2008 and 2015, respectively. He is currently the Coordinator of the Group for Vision and Intelligent Systems (GVIS), whose objective is to research and develop solutions to problems related to cybersecurity and cybercrime for the Spanish National Cybersecurity Institute (INCIBE; https://www.incibe.es/en), using artificial intelligence. His current research interests include natural language processing, computer vision, and machine and deep learning.

ENRIQUE ALEGRE received the M.Sc. degree in electrical engineering from the University of Cantabria, in 1994, and the Ph.D. degree from the University of León, Spain, in 2000. Currently, he is the Head of the Research Group for Vision and Intelligent Systems (GVIS) and an Associate Professor with the Department of Electrical, Systems, and Automation, University of León. His research interests include computer vision and pattern recognition applications to medical and industrial problems and, more recently, machine learning and computer vision applications for crime control and prevention.