future internet

MDPI

Article

Botnet Detection Based On Machine Learning Techniques Using DNS Query Data

Xuan Dau Hoang [1,*] and Quynh Chi Nguyen [2]

[1] Posts and Telecommunications Institute of Technology, Hanoi 100000, Vietnam
[2] Samsung SVMC, Hanoi 100000, Vietnam; nguyenquynhchi2204@gmail.com
* Correspondence: dauhx@ptit.edu.vn; Tel.: +84-904-534-390

Received: 26 April 2018; Accepted: 16 May 2018; Published: 18 May 2018

Abstract: In recent years, botnets have become one of the major threats to information security because they have been constantly evolving in both size and sophistication. A number of botnet detection measures, such as honeynet-based and Intrusion Detection System (IDS)-based, have been proposed. However, IDS-based solutions that use signatures seem to be ineffective because recent botnets are equipped with sophisticated code update and evasion techniques. A number of studies have shown that abnormal botnet detection methods are more effective than signature-based methods because anomaly-based botnet detection methods do not require pre-built botnet signatures and hence they have the capability to detect new or unknown botnets. In this direction, this paper proposes a botnet detection model based on machine learning using Domain Name Service query data and evaluates its effectiveness using popular machine learning techniques. Experimental results show that machine learning algorithms can be used effectively in botnet detection and the random forest algorithm produces the best overall detection accuracy of over 90%.

Keywords: botnet detection; botnet detection model; machine learning-based botnet detection; domain generation algorithm botnet detection; fast flux botnet detection

1. Introduction

In recent years, botnets have been considered one of the major security threats among all types of malware operating on the Internet [1,2]. Botnets have been constantly evolving on the global Internet in both scale and sophistication of control techniques. Each botnet member is called a bot. A bot is a malware created by a hacking group (called *botmaster*) that allows them to control infected computer systems remotely. Bots are different from other forms of malware in that they are highly autonomous and are equipped with the ability to use communication channels to receive commands and code updates from their control system. They can also notify their working status to their control system periodically. The botnet control system, or Command & Control (C & C) servers, is the means by which the botmaster sends commands and code updates to bots. Botnets are often used to transmit malware, send spams, steal sensitive information, deceive, generate virtual clicks, or more seriously to carry out large-scale network attacks, such as DDoS attacks. According to some security reports, about 80% of the Internet traffic is related to the activities of botnets, including spamming and network attacks [1,2].

Domain Name Service (DNS) is an essential service on the Internet that allows the resolution of hostnames, or domain names to Internet Protocol (IP) addresses and vice versa. For example, every time a web client browser accesses a web page, it first sends a request to the DNS system to find the IP address of the web server, and then it uses the IP address to access the web server and load the requested web page. As such, most legitimate applications use DNS services when making requests for accessing network services. However, DNS services are also used by bots of botnets as legitimate applications. Bots send DNS queries to find the IP address of the C & C server and when they have an

IP address, they access the C & C server to receive commands, as well as to download the updated bot code. To evade the scanning and detecting of C & C servers, the botmaster is constantly changing the names and IP addresses of C & C servers using predefined techniques, such as Domain generation algorithms (DGA), or Fast flux (FF) [3,4]. The names and IP addresses of C & C servers are constantly being pushed to the DNS system. Bots are also equipped with the ability to automatically generate C & C server names in accordance with these techniques. As a result, bots can still find IP addresses of C & C servers by generating their hostnames automatically and using these hostnames to query the DNS service. Therefore, monitoring and analyzing DNS query data can reveal the existence of malicious activities in the monitored network, since some of the DNS query data may be generated by botnets. An in-depth analysis of suspicious DNS queries may reveal valuable information regarding the presence of C & C servers and botnets.

This paper examines and evaluates the effectiveness of the bonnet detection method using DNS query data based on a number of commonly used machine learning techniques, including kNN, decision trees, random forest and Naïve Bayes. Based on the evaluation results, we propose a bonnet detection model based on machine learning using DNS query data. The rest of the paper is structured as follows: Section 2 presents related works and Section 3 first introduces some common machine learning techniques and then describes the machine learning based botnet detection model. Section 4 describes experimental scenarios, results and the evaluation and Section 5 is our conclusion.

2. Related Works

Due to the robust development of botnets and the information security risks that they may cause, the detection of botnets in the networks is a hot research topic. Currently, there are two main directions in the botnet detection research, including (1) honeynet-based techniques and (2) IDS-based techniques [3,4]. The techniques in direction (1) usually build honeynets to collect botnet information, and then analyze characteristics and behaviors of botnets. In general, honeynet-based botnet detection systems have the advantages of being easy to build and requiring less computing resources. However, these systems are often limited in their ability to scale and interact with malicious behaviors. In addition, hackers can use honeynets to develop new evasive techniques. The direction (2) used in the botnet detection is based on the IDS. An IDS is a software application, or a hardware device that monitors networks or systems to detect malicious behaviors, or the violation of security policies and to notify the administrators. In particular, IDSs are usually installed to monitor data packets transmitted over a network gateway, or events occurring on a system, and then to conduct the analysis to look for the signs of a botnet. IDS-based botnet detection techniques are divided into two groups: signature-based detection and anomaly-based detection. In the anomaly-based IDS botnet detection group, the botnet detection research based on the analysis of DNS queries is one of the most promising approaches [3,4]. The following part of this section describes some typical solutions.

The proposed solutions for detecting botnets based on the collection and analysis of DNS queries have been published quite abundantly. In particular, Ramachandran et al. [5] proposed counter-intelligence techniques for detecting botnet members by monitoring DNS blacklist (DNSBL) database queries from botnet owners to determine if bot members are on the sending spam blacklist. Research results shows that the proposed techniques are capable of accurately detecting botnets' spam bot members.

An interesting approach to discovering botnets is to detect botnets by monitoring queries from bots sending to DNS systems to find IP addresses of C & C servers to download the commands and bot updated code. By monitoring DNS requests, one can detect the existence of bots and botnets. In this direction, Villamari-Salomo et al. [6] proposed the botnet identification techniques based on the anomaly detection by monitoring DNS queries. The proposed approach attempts to detect domain names with very high query rates and non-existence domain names. This is feasible because there are a large number of bots that query the DNS service for IP addresses of C & C servers at the same period of time, and many of them have not yet been updated so they still query the terminated domains.

More broadly, Perdisci et al. [7] proposed a method for monitoring recursive DNS queries to detect malicious services related to botnet-controlled computers, such as junk blogging, junk messaging and spamming. The test results on 2 actual Internet Service Provider (ISP) networks in 45 days suggest that the proposed method accurately detects the malicious services.

Regarding the monitoring of DNS queries for botnet detection, there have been some proposed solutions for detecting domain names that are automatically generated using algorithms, or that are maliciously used. Accordingly, Yadav et al. [8] proposed a method for distinguishing algorithm generated domain names commonly used in botnets and legitimate domain names using the character distribution in the domain names. Similarly, Stalmans et al. [9] used the C5.0 decision tree method and Bayesian statistics to classify automatically generated domain names with legitimate domain names. Also for the purpose of detecting domain names associated with botnets, Antonakakis et al. [10] proposed a novel detection system, called Kopis. Kopis monitors high-level DNS queries in the DNS hierarchy and analyzes the patterns of global DNS query responses for detecting malware-related domain names. The Kopis's test results show that the system achieves more than 98% accuracy and under 0.5% false detection rates. Kopis also has the ability to detect malware-related domain names many days before they are included in the public blacklists.

Using a different approach, Bilge et al. [11] introduced the EXPOSURE system that allows for the extensive monitoring of DNS traffic to detect domain names that are associated with malicious behaviors. EXPOSURE uses 15 features to distinguish suspicious domain names from legitimate domain names. The test results show that the system is well scalable and can recognize new domain names related to malicious behaviors, as they are used for C & C servers, for spamming and for phishing websites. For the purpose of detecting suspicious and malicious behaviors originating from botnets, Jiang et al. [12] proposed a method for identifying suspicious behaviors based on the analysis of failed DNS queries using a tool called DNS Failure Graphs. DNS failure queries are queries with non-existent domain names, expired domain names, or errors in the DNS system. This suggestion comes from the fact that suspicious behaviors triggered by spamming, trojans' activities, especially bot activities include DNS lookup queries for IP addresses of mail servers, or C & C servers. A significant number of these queries are DNS lookup queries that fail because of the non-existence domain names or the expired domain names. Test results on a three-month DNS query dataset of a large network indicate that the proposed method can identify new behavioral abnormalities with a high probability of originating from unknown bots.

In the opposite direction, Kheir et al. [13] proposed a system called Mentor that allows the removal of legitimate domain names from the blacklist of domain names of C & C servers using a positive reputation-based DNS system. Mentor uses a crawler and statistical characteristics of site content and domain names to categorize legitimate domain names and suspicious domain names based on supervised learning. Experimental results on some botnet domain blacklists show that the system is capable of accurately identifying legitimate domain names with low error rates.

In this paper, we extend the proposals in [8,9,14] by examining and evaluating the performance of the bonnet detection method using DNS query data based on several supervised machine learning techniques, including kNN, decision trees, random forest and Naive Bayes. Based on the survey results, we propose the botnet detection model based on machine learning using DNS query data.

3. Machine Learning Based Botnet Detection

3.1. Introduction to Machine Learning and Its Techniques

3.1.1. Introduction to Machine Learning

Machine learning is a field of computer science, involving the study and construction of techniques that enable computers to self-study based on the input data to solve specific problems [15,16]. Based on the learning methods, machine learning techniques are usually divided into three main categories: supervised learning, unsupervised learning and semi-supervised learning [16]. Supervised learning is

a form of learning in which training data is labeled. The machine will "learn" from the labeled patterns to build the classifier and use it to predict labels for new data. In contrast, unsupervised learning is a form of learning in which training data has not been labeled. In this form, the machine will "learn" by analyzing the data characteristics to construct the classifier. Semi-supervised learning is the hybrid approach between supervised and unsupervised learning. In semi-supervised learning, the input training dataset will contain both labeled and unlabeled data. Each method has its own advantages and disadvantages and has its own application domain. In this paper, we only examine the effectiveness of supervised learning techniques in botnet detection and the next subsection briefly describes some of the common supervised machine learning algorithms, including k-nearest neighbor (kNN), decision tree, random forest, and Naive Bayes. Interested readers may find the full details of these machine learning algorithms in [15,16].

3.1.2. Common Supervised Machine Learning Techniques

3.1.2.1. kNN

kNN (k-Nearest Neighbor) is one of the simplest supervised machine learning algorithms. The idea of kNN is that it will classify the new object based on its k nearest neighbors, where k is a predefined positive integer [16]. kNN algorithm is comprehensively described in the following steps:

- Step 1: Determine the parameter value k.
- Step 2: Calculate the distance between the new object that needs to be classified with all objects in the training dataset.
- Step 3: Arrange computed distances in the ascending order and identify k nearest neighbors with the new object.
- Step 4: Take all the labels of the k neighbors selected above.
- Step 5: Based on the k labels that have been taken, the label that holds the majority will be assigned to the new object.

3.1.2.2. Decision tree

Decision tree is a prediction model that is a mapping from observations of a thing, or a phenomenon to the conclusions about the objective value of things, or phenomena. Decision tree creates models that allow the classification of an object by creating a set of decision rules. These rules are extracted based on the set of characteristics of the training data. In a decision tree, leaves represent classes and each child node in the tree and its branches represent a combination of features that lead to classification. Thus, classifying an object will begin with checking the value of the root node, and then continuing downward under the tree branches corresponding to those values. This process is performed repeatedly for each node until it cannot go any further and touch the leaf node. For the best model, the decision to select the root node and sub-node while building the decision tree is based on the Information Gain (IG) and Gini Impurity measurements [16]. The decision tree algorithm used for experiments in this paper is C4.5.

3.1.2.3. Random forest

Random forest (RF) is a member of the chain of decision tree algorithms. The idea of this algorithm is to create some decision trees. These decision trees will run and give independent results. The answer predicted by the largest number of decisive trees will be chosen by the random forest [16]. To ensure that the decision trees are not the same, random forest randomly selects a subset of the characteristics of each node. The remaining parameters are used in the random forest as those in the decision trees.

3.1.2.4. Naïve Bayes

Naive Bayes is a conditional probability model based on the Bayes theorem [16]. In the classification problem, the data includes: D is the set of training data that has been vectorized as $\vec{x} = (x_1, x_2, ..., x_n)$, C_i is subclass i, with $i = \{1, 2, ..., m\}$. Assume that all properties are conditionally independent of each pair together. According to Bayes theorem, we have:

$$P(C_i|\mathrm{X}) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (1)$$

Based on the conditionally independent characteristic, we have:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \quad (2)$$

where, $P(C_i \mid X)$ is the probability of class i given sample X, $P(C_i)$ is the probability of class i and $P(x_k \mid C_i)$ is the probability of the *kth* property that has the value of x_k given X in class i.

3.2. Proposed Botnet Detection Model Based on Machine Learning

Figure 1 describes the proposed botnet detection model based on machine learning using DNS query data. The model is built on the analysis in Section 1 that bots of botnets routinely send lookup queries to the DNS system to find IP addresses of C & C servers using automatically generated domain names. The detection model is implemented in two phases: (a) the training phase and (b) the detection phase. During the training phase, the DNS query data is collected, and then domain names in DNS queries are extracted. Next, the set of domain names is pre-processed to extract the features for the training. In the training phase, machine learning algorithms are used to learn the classifiers. Through the evaluation process, the machine learning algorithm that gives the highest overall classification accuracy will be selected for use in the proposed detection model. During the detection phase of the model, the DNS queries are monitored and passed through the process of extracting the domain names, pre-processing, and classifying using the classifier produced from the training phase to determine if a domain name is legitimate, or a botnet domain name. The pre-processing step for each domain name in the training and detection phase is the same. However, this step is done in the offline mode for all the domain names of the training dataset in the training phase while it is done for each domain name extracted from the DNS query on the fly in the detection phase.

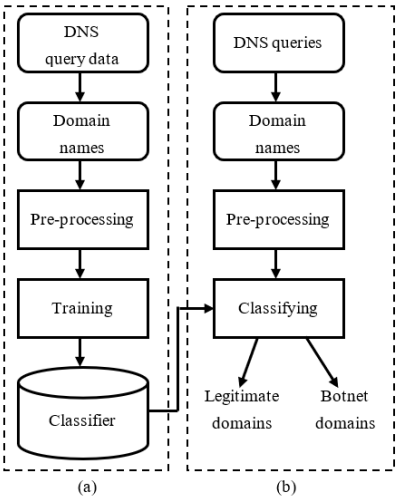

Figure 1. Proposed botnet detection model based on machine learning using Domain Name Service (DNS) query data: (**a**) training phase and (**b**) detection phase.

4. Experiment and Evaluation

4.1. Experimental Dataset

To evaluate the classification performance of botnet domain names using machine learning algorithms, we use the extracted and labeled domain name datasets, which include a set of benign domain names and a set of malicious domain names used by botnets. The set of benign domains includes 30,000 top domain names ranked by Alexa. Benign domain names are checked at the *virustotal.com* website to make sure they are really benign. The set of malicious domain names collected at [17,18] includes 30,000 domain names created by the Conficker botnet and the DGA botnet. From the original datasets, the domain names are processed to remove the top-level domain. For example, with the domain name "example.com", after processing, the resulting data is "example".

4.2. Data Pre-Processing

According to [8,9,14], automatically generated botnet domain names often have DNS characteristics, network characteristics and lexical features that are different from those of legitimate domain names. Yadav et al. [8] proposed to use the distribution analysis of vowels, numbers, and other characters in domain names to distinguish legitimate domain names and domain names generated by botnet algorithms. More broadly, Stalmans et al. [9] proposed to use two groups of domain names' characteristics, including DNS characteristics (IP address, network address, etc.) and lexical features (the character distribution of the domain names). Meanwhile, da Luz [14] proposed to use 36 characteristics in two groups, including 18 vocabulary characteristics (mean, variance and standard deviation of 1-g, 2-g, 3-g and 4-g; entropy; characteristics of characters, numbers, vowels and consonants) and 18 network characteristics (TTL, number of network addresses, etc.). In this paper, we focus on exploiting the statistical vocabulary characteristics of 2-g and 3-g clusters, and the vowel distribution characteristics of domain names. Specifically, we use 16 statistical vocabulary features of 2-g (bi-gram) and 3-g (tri-gram) clusters, and 2 vowel distribution characteristics to extract 18 features of each domain name for the training phase and also for the detection phase.

4.2.1. Features of Bi-gram and Tri-gram Clusters

Bi-gram is a cluster of two adjacent characters extracted from a string. For example, the domain name "example" (top level domain ".com" was removed) includes the following bi-grams: *ex*, *xa*, *am*, *mp*, *pl* and *le*. A domain name can contain characters in 26 alphabetical characters (a–z), numeric characters (0–9), "." and "-" characters. Therefore, the total number of possible bi-grams, *TS*(bi-gram) is $38 \times 38 = 1444$. From the set of benign domain names, we extract a list of N most frequently encountered bi-grams, denoted as *DS*(bi-gram). *DS*(bi-gram) is used for calculating 8 bi-gram related features of each domain name.

Tri-gram is a cluster of three adjacent characters extracted from a string. For example, the domain name "example" consists of the following tri-grams: *exa*, *xam*, *amp*, *mpl* and *ple*. Similar to the calculation of the total number of bi-grams, the total number of possible tri-grams, *TS*(tri-gram) is $38 \times 38 \times 38 = 54{,}872$. From the set of benign domain names, we extract the list of M most frequently occurring tri-grams, denoted as *DS*(tri-gram). *DS*(tri-gram) is used for calculating 8 tri-gram related features of each domain name.

The features of bi-grams and tri-grams (we call both of them as *n-gram*) for each domain name d include:

- *count(d)* is the number of n-grams of the domain name d that are also found in *DS*(*n-gram*).
- $m(d)$ is the general frequency distribution of the n-grams of the domain name d, which is calculated by the following formula:

$$m(d) = \sum_{i=1}^{count(d)} f(i) \times index(i) \quad (3)$$

where $f(i)$ is the total number of occurrences of n-gram i in $DS(n\text{-}gram)$ and $index(i)$ is the rank of n-gram i in $TS(n\text{-}gram)$.

- $s(d)$ is the weight of n-grams of the domain name d, calculated by the following formula:

$$s(d) = \frac{\sum_{i=1}^{count(d)} f(i) \times vt(i)}{count(d)} \tag{4}$$

where $vt(i)$ is the rank of n-gram i in $DS(n\text{-}gram)$.

- $ma(d)$ is the average of the general frequency distribution of the n-grams of the domain name d, which is calculated by the following formula:

$$ma(d) = m(d)/len(d) \tag{5}$$

where $len(d)$ is total number of n-grams in the domain name d.

- $sa(d)$ is the average of the weight of n-grams of the domain name d, calculated by the following formula:

$$sa(d) = s(d)/len(d) \tag{6}$$

- $tan(d)$ is the average number of popular n-grams of the domain name d, calculated by the following formula:

$$tan(d) = count(d)/len(d) \tag{7}$$

- $taf(d)$ is the average frequency of popular n-grams of the domain name d, calculated by the following formula:

$$taf(d) = \frac{\sum_{i=1}^{count(d)} f(i)}{len(d)} \tag{8}$$

- $entropy(d)$ is the entropy of the domain name d, calculated by the following formula:

$$entropy(d) = -\sum_{i=1}^{count(d)} \frac{vt(i)}{L} \times log(\frac{vt(i)}{L}) \tag{9}$$

where L is the number of popular n-grams in the set of benign domain names, $L = N$ for bi-grams and $L = M$ for tri-grams.

4.2.2. Vowel Distribution Features

According to [8,9], benign domain names often have higher vowel numbers than that of automatically generated domain names by botnets, because benign domain names are often built on grammatical features, while botnet automatically generated domain names usually consist of random characters and do not have meanings. Based on this idea, 2 vowel features of each domain name are built, including:

- $countnv(d)$ is the number of vowels of the domain name d.
- $tanv(d)$ is the average number of vowels of the domain name d, calculated by the following formula:

$$tanv(d) = countnv(d)/len(d) \tag{10}$$

4.3. Experimental Scenarios and Classification Measures

4.3.1. Experimental Scenarios

After the pre-processing process, 18 features extracted from each domain name form a record in the experimental dataset. We selected three training datasets, namely T1, T2 and T3, and a testing dataset, called TEST from the experimental dataset. The T1 set contains 10,000 records from benign domain names and 10,000 records from malicious domain names. The T2 set contains 5000 records from benign domain names and 15,000 records from malicious domain names and the T3 set contains 15,000 records from benign domain names and 5000 records from malicious domain names. The selection of 3 training datasets with different numbers of benign and malicious domain names to assess their effect on detection performance. The TEST dataset includes 10,000 records from benign domain names and 10,000 records from malicious domain names. Records from benign domain names are labeled "good" and records from malicious domain names are labeled "bad". Data for testing is not part of the training sets. Table 1 describes the dimensions of the components of the training datasets and the testing dataset.

Table 1. The training datasets and the testing dataset.

Datasets for Training and Testing	The Number of Records From	
	Dataset Labeled "Good"	Dataset Labeled "Bad"
T1 training dataset	10,000	10,000
T2 training dataset	5000	15,000
T3 training dataset	15,000	5000
TEST testing dataset	10,000	10,000

4.3.2. Classification Measures

The classification measures used in our experiments include PPV (Positive predictive value), FPR (False positive rate), TPR (True positive rate), ACC (Accuracy) and F1 (F1 measure). These measures are computed using the following formulas:

$$\text{PPV} = \frac{TP}{TP + FP} \times 100\% \tag{11}$$

$$\text{FPR} = \frac{FP}{FP + TN} \times 100\% \tag{12}$$

$$\text{TPR} = \frac{TP}{TP + FN} \times 100\% \tag{13}$$

$$\text{ACC} = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \tag{14}$$

$$\text{F1} = \frac{2TP}{2TP + FP + FN} \times 100\% \tag{15}$$

where, *TP* (True Positives) is the number of records labeled "bad" that are classified correctly, *TN* (True Negatives) is the number of records labeled "good" that are classified correctly, *FP* (False Positives) is the number of records labeled "good" that are misclassified to "bad" and *FN* (False Negatives) is the number of records labeled "bad" that are wrongly classified as "good". Table 2 provides the confusion matrix of *TP*, *TN*, *FP*, and *FN*.

Table 2. The confusion matrix.

		Actual Class	
		Bad	**Good**
Predicted class	Bad	TP (True Positives)	FP (False Positives)
	Good	FN (False Negatives)	TN (True Negatives)

For the kNN machine learning algorithm, the model is tested with k = {from 1 to 20} with two cases of non-weighted and weighted neighbors. In particular, the weight is assigned based on the principle: the smaller the distance the greater the weight. Based on the test results, we determine the coefficient k that gives the highest classification accuracy.

For the random forest algorithm, the general prediction result is based on the predictions of the decision trees. Therefore, the number of decision trees that may affect the accuracy of the algorithm. The model was tested with the number of decision trees of {from 5 to 50} to determine the number of trees that produce the highest classification accuracy.

4.4. Experimental Results and Comments

4.4.1. Experimental Results

The model's experimental results using 4 machine learning algorithms on the T1, T2, T3 training datasets and the TEST testing dataset described in Section 4.3.1 show that the kNN algorithm's overall classification accuracy is better with weighted neighbors than with non-weighted neighbors. The k value that gives the best classification performance is not stable and it depends on the training sets. The best classification performance using the kNN algorithm with k values of 13, 9 and 15 on the T1, T2 and T3 training sets, respectively.

Similar to kNN, the number of decision trees of the random forest algorithm that produces the best classification results is also unstable and it depends on the training sets. The random forest algorithm gives the best classification performance with the numbers of decision trees of 30, 38 and 27 on the T1, T2 and T3 training datasets, respectively. Tables 3–5 provide the model's testing results using 4 machine learning algorithms on the T1, T2 and T3 training sets, respectively.

Table 3. Classification performance of the detection model using T1 training set.

Machine Learning Algorithms	**Measures (%)**				
	PPV	**FPR**	**TPR**	**ACC**	**F1**
kNN (k = 13)	89.50	10.70	91.00	90.20	90.30
C4.5	89.10	11.10	91.20	90.10	90.10
RF (30 trees)	90.70	9.30	91.00	90.80	90.80
Naïve Bayes	83.10	18.30	90.20	85.90	86.50

Table 4. Classification performance of the detection model using T2 training set.

Machine Learning Algorithms	**Measures (%)**				
	PPV	**FPR**	**TPR**	**ACC**	**F1**
kNN (k = 9)	82.70	20.10	96.40	88.10	89.00
C4.5	81.50	22.10	97.30	87.60	88.70
RF (38 trees)	84.20	18.10	96.60	89.20	89.96
Naïve Bayes	82.80	18.80	90.50	85.80	86.50

Table 5. Classification performance of the detection model using T3 training set.

Machine Learning Algorithms	Measures (%)				
	PPV	**FPR**	**TPR**	**ACC**	**F1**
kNN (k = 15)	94.10	5.00	80.40	87.70	86.70
C4.5	93.40	5.70	80.90	87.60	86.70
RF (27 trees)	94.40	4.80	80.90	88.10	87.20
Naïve Bayes	83.90	17.10	89.10	86.00	86.40

4.4.2. Comments

From the experimental results given in Tables 3–5, we can see that the Naive Bayes algorithm produces the lowest overall classification accuracy (ACC) and the random forest algorithm produces the highest classification accuracy among 4 machine learning algorithms. However, the differences in the classification accuracy among algorithms are not so great. Two machine learning algorithms, including kNN with weighted neighbors and C4.5 decision trees have roughly the same overall classification accuracy. For the kNN algorithm, the choice of parameter k plays an important role in the classification performance and is usually determined empirically. The Naive Bayes algorithm provides the lowest overall classification accuracy, but it has the advantages of being simple and low time and computation requirements for training and testing.

It can also be seen that the random forest algorithm yields significantly better results than the C4.5 decision tree algorithm. However, the training time of the random forest is longer due to the number of trees requiring training is more. However, the training of the random forest classifier can be done offline, so it does not affect the classification speed in the testing period. Thus, the random forest machine learning algorithm has the highest overall classification accuracy and is selected for implementation in the proposed botnet detection model.

In addition, the ratio between the number of benign domain names and the number of malicious domain names has a significant impact on classification measures. Accordingly, the T1 training set gives the highest ACC and F1 measures. However, its FPR is relatively high. The ACC and F1 measures produced by the T3 training set are not high, but the FPR is relatively low. At the same time, the T2 set gives the worst results of low ACC and F1 measures and very high FPR measure. As such, the ratio of the number of benign domain names and the number of malicious domain names to be chosen as 1:1 (the T1 dataset) will produce the best classification results.

5. Conclusions

In this paper, we investigated the effectiveness of the botnet detection model based on machine learning techniques using DNS query data. The experimental results on DGA botnet and FF botnet datasets show that most of the machine learning techniques used in the model achieved the overall classification accuracy over 85%, among which the random forest algorithm gives the best results with the overall classification accuracy of 90.80%. Based on this result, we propose to select the random forest algorithm for the proposed botnet detection model using DNS query data.

In the future, we continue to test the proposed model with larger datasets and analyze the effects of the domain name features on the detection accuracy, as well as research and propose new features to improve the detection accuracy of the proposed model.

Author Contributions: X.D.H. raised the idea, initialized the project and designed the experiments; Q.C.N. carried out the experiments under the supervision of X.D.H.; Both authors analyze the data and results; X.D.H. wrote the paper.

Funding: This research was funded by the Ministry of Science and Technology, Vietnam grant number KC.01.05/16-20.

Acknowledgments: This work has been supported by the CyberSecurity Lab, Posts and Telecommunications Institute of Technology, Hanoi, Vietnam and funded by the Ministry of Science and Technology, Vietnam grant number KC.01.05/16-20.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Authority of Information Security. *The 2016 Vietnam Information Security Report*; Authority of Information Security, MIC: New York, NY, USA, 2016.
2. Ferguson, R. The History of the Botnet. Available online: http://countermeasures.trendmicro.eu/the-history-of-the-botnet-part-i/ (accessed on 1 February 2018).
3. Alieyan, K.; Almomani, A.; Manasrah, A.; Kadhum, M.M. A survey of botnet detection based on DNS. *Nat. Comput. Appl. Forum* **2017**, *28*, 1541–1558. [CrossRef]
4. Li, X.; Wang, J.; Zhang, X. Botnet Detection Technology Based on DNS. *J. Future Internet* **2017**, *9*, 55. [CrossRef]
5. Ramachandran, A.; Feamster, N.; Dagon, D. Revealing botnet membership using DNSBL counter-intelligence. In Proceedings of the 2nd USENIX: Steps to Reducing Unwanted Traffic on the Internet, San Jose, CA, USA, 7 July 2006; pp. 49–54.
6. Villamari-Salomo, R.; Brustoloni, J.C. Identifying botnets using anomaly detection techniques applied to DNS traffic. In Proceedings of the 5th IEEE consumer communications and networking conference (CCNC 2008), Las Vegas, NV, USA, 10–12 January 2008; pp. 476–481.
7. Perdisci, R.; Corona, I.; Dagon, D.; Lee, W. Detecting malicious flux service networks through passive analysis of recursive DNS traces. In Proceedings of the Annual Computer Security Applications Conference, Honolulu, HI, USA, 7–11 December 2009; pp. 311–320.
8. Yadav, S.; Reddy, A.K.K.; Reddy, A.; Ranjan, S. Detecting algorithmically generated malicious domain names. In Proceedings of the 10th ACM sigcomm Conference on Internet Measurement, Melbourne, Australia, 1–30 November 2010; pp. 48–61.
9. Stalmans, E.; Irwin, B. A framework for DNS based detection and mitigation of malware infections on a network. In Proceedings of the 2011 Information Security for South Africa (ISSA), Johannesburg, South Africa, 15–17 August 2011; pp. 1–8.
10. Antonakakis, M.; Perdisci, R.; Lee, W.; Vasiloglou, N., II; Dagon, D. Detecting malware domains at the upper DNS hierarchy. In Proceedings of the USENIX security symposium, San Francisco, CA, USA, 8 August 2011; p. 16.
11. Bilge, L.; Kirda, E.; Kruegel, C.; Balduzzi, M. *Exposure: Finding Malicious Domains Using Passive DNS Analysis*; NDSS: New York, NY, USA, 2011.
12. Jiang, N.; Cao, J.; Jin, Y.; Li, L.; Zhang, Z.L. Identifying suspicious activities through DNS failure graph analysis. In Proceedings of the18th IEEE International Conference on Network Protocols (ICNP), Kyoto, Japan, 5–8 October 2010; pp. 144–153.
13. Kheir, N.; Tran, F.; Caron, P.; Deschamps, N. Mentor: positive DNS reputation to skim-off benign domains in botnet C&C blacklists. In *ICT Systems Security and Privacy Protection*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 1–14.
14. Da Luz, P.M. *Botnet Detection Using Passive DNS*; Radboud University: Nijmegen, The Netherlands, 2014.
15. Sangani, N.K.; Zarger, H. *Machine Learning in Application Security. Advances in Security in Computing and Communications*; IntechOpen: Karnataka, India, 2017.
16. Smola, A.; Vishwanathan, S.V.N. *Introduction to Machine Learning*; Cambridge University Press: Cambridge, UK, 2008.
17. Conficker. Available online: http://www.cert.at/static/conficker/all_domains.txt (accessed on 10 November 2017).
18. DGA Dataset. Available online: https://github.com/nickwallen/botnet-dga-classifier/tree/master/data (accessed on 10 November 2017).