Detection of Botnets That Use Algorithmically Generated Domains Using Machine Learning

Noshin Nawar Sadat

David R. Cheriton School of Computer Science
University of Waterloo
nnsadat@uwaterloo.ca

***Abstract*—Detecting bots based on the detection of domain names generated by Domain Generation Algorithms (DGAs) was the main goal of this project. To this aim, existing research works were studied and on the basis of information derived from their results, an attempt was made to build a classification model that separated DGA generated domain names from legit domain names. A different technique to bolster the classifier's output was proposed. However, due to lack of appropriate data set, it was not possible to test that technique.**

***Index Terms*—botnet detection, DGA, machine learning**

I. Introduction

Internet has integrated into our daily life to such an extent that it has become impossible for us to function without using it even for a day. This has also, unfortunately, led to the rise in the number of some miscreants who exploit the internet to harm people and businesses, and to make illegal money. One of the most common ways that these online criminals use to achieve their goals is the use of botnets. A botnet is a collection of compromised computers, also known as bots [1], which are leveraged to perform large scale online criminal activities, such as, spam, phishing, information theft, Distributed Denial Of Service (DDOS) attacks, etc [2].

To create a botnet, the hacker, also known as the botmaster, first sends malware to vulnerable hosts in the disguise of email attachments or downloadable files. Once this malware is downloaded into the host, it starts to try to communicate with the command and control (C&C) server which is used by the botmaster to give out instructions. If the C&C servers have the same domain name, then they can be easily detected and blacklisted. To avoid this, botmasters use Domain Generation Algortihms (DGAs) to generate random domain names and register them against their C&C server IPs [4]. The malwares in the compromised hosts also run the same DGAs to generate numerous domains and check if any of those domains belong to their C&C server by generating DNS queries for those domains and then contacting them with the resolved IP addresses. Once they find their C&C server, they start communicating with it to receive instructions.

The time when the DGA botnet tries to locate its C&C server by making DNS queries is the perfect opportunity to tackle their communication with the C&C servers. From the domain names received by a DNS server for resolution, if it is possible to identify which ones are DGA generated, then not only can the bots of a botnet be hunted down but also, the C&C servers can be tracked by monitoring those botnets.

Due to the ever rising threats of botnet attacks, numerous research work have been initiated in order to detect botnets. Most of the studied research works considered a certain time window in order to collect DNS traffic data and then performed analysis on them to detect botnets [2] [5] [6]. If a DGA bot sends DNS query requests at time intervals greater than these time windows, then they can easily bypass these detection system.

Although it would have been better if a system could simply take a look at a domain name and give a decision on whether it is a DGA generated domain name or not, it is not really possible because the host generating a DNS query which is similar to those generated by DGAs might not actually be infected by a bot. The user might have mistakenly or intentionally entered such a domain name. Hence, it is important to

also consider botnet like behavior, such as, multiple hosts asking for the same non-existent domain name resolution or the same host asking for multiple non-existent domain name resolutions where the domains have similar lexical features. However, there might be just one bot in a network and it might not be possible to collect data on multiple hosts sending the same query.

Considering such situations, it has been planned to build a DGA botnet detection system which will include the following modules:

- *Data Collection and Pre-processing* : Instead of using fixed time windows to collect data, the module will use a certain limit N of non-existent domain queries per IP address. Every time, the module comes across the same IP address sending a non-existent domain query, it will increment the count until it reaches N. Whenever the N count is reached for an IP, the domain names queried by that IP will be sent into the Classification module.
- *Classification* : In this module, each of the N domains will go through a classifier model. If out of the N domains, a certain number x of domains are classified as DGA generated, then the corresponding IP will be marked as a bot.

Due to lack of appropriate data sets as well as time constraints, which will be discussed more in the following sections, it was not possible to implement and test the full system. A classifier model was built, however, for the classification module and it was tested by using two data sets.

The outline of this paper is as follows:

- Section II is a review of previous works on detection of DGA botnets
- Section III describes how the classifier model was eventually built
- Section IV shows the analysis of the classifier model by testing it with a data set
- Section V discusses the possible future explorations as well as their expected contributions
- Section VI summarizes the whole paper and gives conclusive remarks about the project

II. LITERATURE REVIEW

The focus of the project is on real time detection of DGA botnets. In this regard, several papers were studied:

[3] identified two main approaches towards botnet detection research - honeynet setup and corresponding data analysis, and monitoring and analysis of passive network traffic. The honeynet approach is mainly useful to understand the signature characteristics and behaviors of botnets, not for botnet detection. The latter approach is used to detect botnets. In this approach, there are many methods for botnet detection. [3] mainly identified four of them:

- Signature based detection where particular characteristics of known botnets are used
- Anomaly based detecion where anomalous behaviors in network traffic, such as, unusually high traffic volume, network latency etc are used
- DNS based detection where the DNS queries sent to a DNS server are used
- Mining based detection where various machine learning techniques are used to detect communication between botnets and the hacker

[5] used behavioral models to detect DGA bots. In order to build their detection models, they collected network traffic of normal hosts as well as malware infected hosts. They created models of each connection of the traffic. Each connection was identified by a four-tuple composed of source IP address, destination IP address, destination port and the protocol. The models were represented by a string of characters. The characters were added into the string based on three features of each flow size, duration and periodicity. Then they trained Markov chain model for each connection. The new incoming traffic data would be collected every five minutes and the connections in the traffic would be used to create strings of characters which would be matched against the trained models to see if they were generated from those models. Their method had pretty good detection rates.

[6] focused on the detection of DGA botnets based on the lexical features of the domain names. In order to do so, they first collected all the non-existent domain names from DNS responses for a suitable amount of time (3minutes). The domain names were then filtered to remove invalid TLDs, white-listed and popular domains and so on. Then they detected those

hosts which produced the highest peaks of unresolved DNS queries within that time. Next, the resolved DNS requests right next to those peaks were collected. From these collected domain names, they extracted five linguistic features number of levels, distances of monogram and bi-gram probability distribution of the second and third levels from those of the English dictionary, entropy of the character distributions for the second and third levels, and the number of character in the second and third levels. They performed k-means clustering on the domain names based on these features. The clusters with higher number of unresolved domains than resolved domains were identified as potentially malicious. They then set an anomaly indicator for each cluster. Its value is proportional to the value of homogeneity of each cluster. This indicator is used to remove false positive. Their method was able to detect almost all the DGA based unresolved domain names and had a really low false positive rate.

[7] worked on identifying DGA generated domain names from a mix of both DGA generated domain and normal domain names using supervised machine learning algorithms. For this purpose, they used a list of DGA domains and a list of Alexa top 1 million domains. They used two groups of algorithms for their research handcrafted feature based (C4.5, SVM, ELM) and implicit feature based (HMM, LSTM, Recurrent SVM, CNN+LSTM, bidirectional LSTM). They used 9 different lexical features for the first group of algorithms length of domain, entropy, dictionary matching score, n-gram normality scores for n=1,2,3,4,5 and 3-,4- and 5-grams merged. From their experiments, they found that recurrent SVM and bidirectional LSTM were the best among the other algorithms in detecting DGA domains.

[8] used a data set of DGA names and one of benign domain names in the training phase of their system. The top level domains were removed from the domain names at first. In order to train their classifiers, they used 8 lexical features of both bi-grams and tri-grams, and 2 vowel distribution characteristics. The bi-gram and tri-gram characteristics used were count, general frequency distribution, weight, average general frequency distribution, average weight, average number of popular n-grams, average frequency of popular n-grams and entropy. The vowel distribution features used were count of vowels in domain name and average number of vowels. They applied kNN, C4.5, random forest and Nave Bayes algorithms to the training data set and in the testing phase, they found random forest to be most accurate in detecting malicious domains.

[2] aimed at detecting CC domain names from passive DNS traffic. They first collected the DNS traffic and then based on the RCODE of the responses, they classified the domain names into active and NXDomains. Moreover, they also considered ISP hijacked DNS request records as NXDomains. Then from the NXDomain list, they omitted out the names that were incompatible with the rules of naming domains and they also removed CDN, cache server, Bittorrent download domains etc. Next they performed bipartite clustering on the NXDomains to single out those domains which had only one IP address requesting it and those IP addresses which had requested only one NXDomain. They then calculated the variance function of the frequency of character occurrence in the domains and if its value was less than a certain threshold, they considered it a NXDomain generated by a DGA. After this step, they used an SVM classifier to classify the Active domains that they had previously separated out. The features they had used for the classifier were entropy of length, entropy of information, readability, mean ranking of n-grams, count of different letters present, count of different digits present, ratio of consecutive letters or numbers and a common score of TLD.

[12] detected the existence of algorithmically generated domain names using statistical measures, namely, namely, Kullback-Leibler divergence, Jaccard index and Levenshtein edit distance. The found the Jaccard index performed the best among them all while Kullback Leibler divergence performed the worst. They were able to detect the presence of COnficker botnet in the data set. They were also able to detect a new kind of malware. They suggested that their system could be used to detect the presence of domain fluxing activities in traffic, which can be further analyzed by network security analyst to confirm the existence of malware.

[15] emphasized that DNS traffic demonstrates time dependencies and thus they aimed at building a machine learning based time series decision trees where they used DNS traffic time series to detect botnets. Although the false positive rates of their model were very high, their decision tree model was able to detect temporal patterns in the data, which they suggested

would complement existing bot detection methods by incorporating the time features.

[16] used frequency distributions of tri-grams and penta-grams of websites from Alexa and some common dictionary words, length of the domain names and entropy of the domain names in order to build their classifier for classifying between legit and DGA domains. The algorithms they tested were Random Forest, Support Vector Machine and Naive Bayes. They found Random Forest to give the best result out of the three algorithms.

[17] emphasized the importance of length of domain names in the classification of DGA and normal domains. They used the length of domain names, the count of vowels in second level domains, the count of consonants in second level domain name, the count of second level domain names, domain trigram conditional probability and domain trigram entropy as features for Random Forest and J48 Decion Tree classifiers. AFter several experimentation, they developed a split model which classified domain names based on features which were governed by the length of the domain name. That is, if the length of the domain name was less than a length threshold *l*, then they considered all features except entropy. Otherwise, entropy was also considered.

III. CLASSIFIER MODEL

The classifier model being the core of the planned system, the first target was to train and build a classifier model. The more accurate the classifier would be, the lesser value of N could be considered.

A. Data set

To train the classifier, both benign domain names and algorithmically generated malicious domain names were required. The initial plan was to use labelled passive DNS traffic data set in order to build a classifier model for the system. However, due to unavailability of such data set, simply lists of both malicious and benign domain names were used. One million domain names were collected from [10] as to represent as benign data set, while 381953 labelled and algorithmically generated domain names were collected from [9].

B. Data Pre-processing

All the top level and second level domain names were removed from the data set. For example, 'google.co.uk' was reduced to 'google'.

Due to the huge sizes of the data sets collected, it took longer processing times. Hence, the sizes of the data sets were reduced. Since the classifier was purely trained on the basis of lexical features of domain names, there was no risk of loosing crucial information by reducing the sizes.

From the benign data set, the first 80,000 domain names were taken, which after removing duplicates, stood to be 79,035 in number.

The malicious data set included domain names generated by 43 malwares between 23-June-2018 to 28-June-2018. 25-June-2018 had data of the most number of different malwares. Hence, only the domain names from that date was taken. To further reduce the size of malicious domain names so that it could be of similar size as the data set of benign domain names, instances of those malwares which generated the least amounts of domain names were removed. Domain names of some of the remaining malwares were halved.

Finally, the two data sets were aggregated and labelled as either 'dga' or 'legit' based on which data set they came from. The final data set for training the classifier looked as shown in table I.

C. Feature Extraction

In order to select a classifier model, the lexical features mentioned in [7] and [8] were all calculated from the data set. Using the chi-square test, the following features were selected from the extracted features and a significant level of accuracy was achieved among the tested algorithms.

1) n-gram Weight of domain d:

$$wt(d) = \frac{\sum_{i=1}^{c(d)} freq(i) \times rank(i)}{c(d)}$$

where, c(d) = number of n-grams in domain d, which are also in the list of n-grams of all benign domains, freq(i) = frequency of occurrence of n-gram i in the list of n-grams of all benign domains, rank(i) = frequency based rank of n-gram i in the list of n-grams of all

TABLE I
TRAINING DATA SET

Alexa (benign)	79035
kraken	8988
tinba	8336
murofet	7130
pykspa	7108
ramnit	7022
ranbyus	6510
shiotab/urlzone/bebloh	6260
Post Tovar GOZ	5500
necurs	5372
qakbot	5000
vawtrak	3150
ramdo	2000
P2P Gameover Zeus	2000
pushdo	1680
chinad	1536
locky	1338
dyre	1333
Cryptolocker-Flashback	1000
Volatile Cedar / Explosive	996
proslikefan	780
sphinx	768
dircrypt	720
fobber	600
tempedreve	249
bamital	240
beebone	210
vidro	200
hesperbot	192
geodo	96
padcrypt	96
symmi	64
corebot	40
Total	165549

benign domains and n = 2, 3.

2) n-gram Normality Score of domain d:

$$ns(d) = \frac{\sum_{i=1}^{t(d)} cnt(i) \times freq(i)}{|d|}$$

where, t(d) = total number of n-grams in domain d and cnt(i) = number of n-gram i in domain d.

3) n-gram Entropy of domain d:

$$e(d) = -\sum_{i=1}^{c(d)} \frac{freq(i)}{l} \times log(\frac{freq(i)}{l})$$

where, l = total number of n-grams in the list of n-grams of all benign domains.

To further increase the accuracy of classifier, deriving ideas from [2], the mean of character frequency of domain names were calculated and plotted as shown in figure 1. All the characters allowed in a domain name by convention were considered, that is, a-z, 0-9, '.' and '-'. The red dotted plot is the character frequency mean of DGA domains and the blue 'x' plot is that of legit domains.

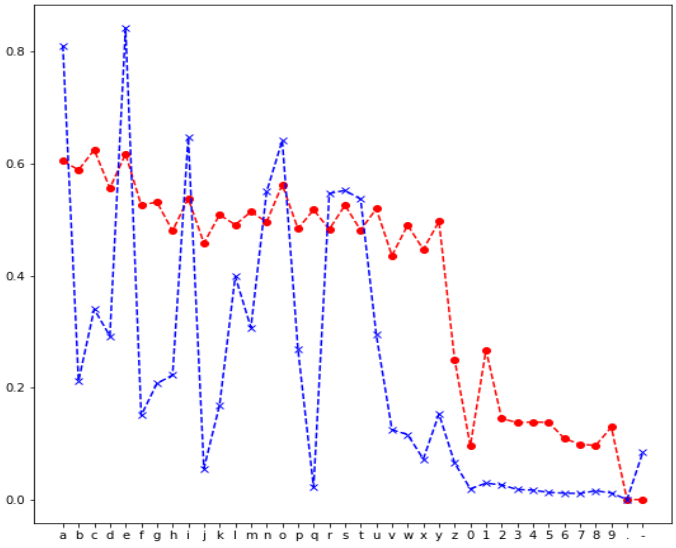

Fig. 1. Mean of character frequency of domain names

Significant difference between the two plots at the numerical characters, the vowels, the '-' sign and the consonants were noticed. Hence, the following feature was calculated per each character for each domain:

$$afreq(d) = \frac{totA(d)}{|d|}$$

where, afreq(d) = frequency of character 'a' in domain d divided by the domain length to normalize it and totA(d) = frequency of character 'a' in domain d. Similarly, bfreq(d), cfreq(d) and so on were calculated for all characters.

In choosing from among these 38 character frequencies, the assumption of [8] was used, where they mentioned that benign domain names tended to have more vowel occurrences than malicious algorithmically generated ones. Hence, along with the previous 3

lexical features (6 in total, considering both bigrams and trigrams)and the frequencies of the vowels, different combinations of the other character frequencies were used. Addition of the frequencies of the numerical characters and the '-' character showed an increase in the accuracies of the tested algorithms, after which the accuracy remained more or less the same no matter what other character frequencies were included.

D. Model Selection

In order to build the classifier, the following machine learning algorithms were compared:

1) Logistic Regression (LR)
2) K Nearest Neighbor (KNN)
3) Decision Trees (DT)
4) Gaussian Naive Bayes (GNB)
5) Support Vector Machine (SVM)
6) Random Forest (RF)

The data set was split randomly into two parts - 80% of it was used for training and comparing the models while the other 20% was used to test the accuracy of the chosen model. 10 fold cross validation was used with the training data set to determine and compare the accuracies of the above mentioned algorithms. The accuracies of the algorithms are shown in table II. Cases of when the character frequencies were used and when they weren't used - both are shown here.

TABLE II
ALGORITHM ACCURACY COMPARISON

	ACCURACY	
ALGORITHMS	*Without frequencies*	*With frequencies*
LR	91.5%	91.6%
KNN (5 neighbors)	87%	87%
KNN (10 neighbors)	87.1%	87.1%
KNN (15 neighbors)	87.2%	87.2%
DT	94.3%	95%
GNB	89.3%	89.3%
SVM	72%	73.5%
RF (10 trees)	95.9%	96.6%
RF (20 trees)	96.1%	96.8%
RF (30 trees)	96.2%	96.9%
RF (40 trees)	96.2	96.9%
RF (50 trees)	96.3	96.9%

Among all the algorithms, Random Forest and Decision trees showed the best results. Hence, both of them were selected to evaluate further in order to chose them as the model classifier. Random forest for different numbers of trees were considered. The algorithms were

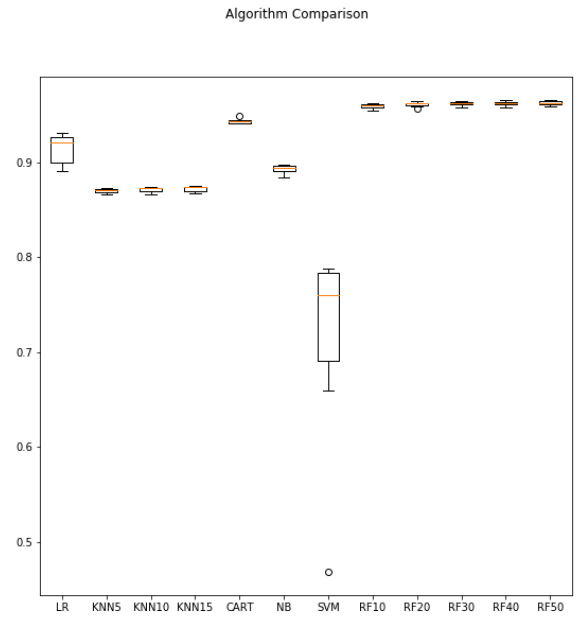

Fig. 2. Comparison of algorithms when no character frequencies considered

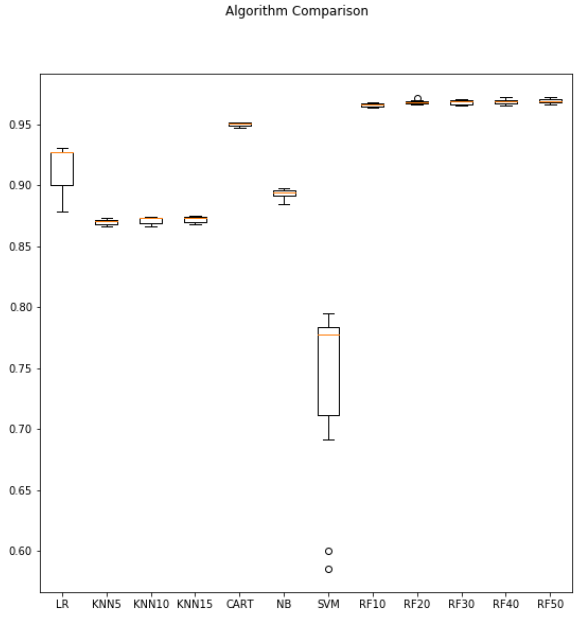

Fig. 3. Comparison of algorithms when character frequencies considered

tested using the testing data set that were separated at the beginning. The confusion matrices, precision, recall, F1 scores and Matthew's correlation co-efficient (MCC) score for each case are shown in the tables III to IX.

TABLE III
CONFUSION MATRIX OF RANDOM FOREST WITH 10 TREES

	PREDICTION	
ACTUAL	*dga*	*legit*
dga	16,795	613
legit	498	15,204

TABLE IV
CONFUSION MATRIX OF RANDOM FOREST WITH 20 TREES

	PREDICTION	
ACTUAL	*dga*	*legit*
dga	16,768	640
legit	419	15,283

TABLE V
CONFUSION MATRIX OF RANDOM FOREST WITH 30 TREES

	PREDICTION	
ACTUAL	*dga*	*legit*
dga	16,769	639
legit	409	15,293

TABLE VI
CONFUSION MATRIX OF RANDOM FOREST WITH 40 TREES

	PREDICTION	
ACTUAL	*dga*	*legit*
dga	16,771	637
legit	401	15,301

TABLE VII
CONFUSION MATRIX OF RANDOM FOREST WITH 50 TREES

	PREDICTION	
ACTUAL	*dga*	*legit*
dga	16,764	644
legit	394	15,308

From the test results, a random forest classifier with 30 trees showed overall good scores. The scores seemed to have saturated from that point onward. Hence, a random forest classifier with 30 trees was selected, given the 22 features, namely, bigram weight, trigram weight, bigram normality score, trigram normality score, bigram entropy, trigram entropy and character frequencies of vowels, numerical characters and '-'.

TABLE VIII
CONFUSION MATRIX OF DECSION TREES

	PREDICTION	
ACTUAL	*dga*	*legit*
dga	16,628	780
legit	817	14,885

TABLE IX
SCORES COMPARISON

ALGORITHMS	PRECISION	RECALL	F1 SCORE	MCC
RF(10 trees)	0.97	0.97	0.97	0.93
RF(20 trees)	0.97	0.97	0.97	0.94
RF(30 trees)	0.97	0.97	0.97	0.94
RF(40 trees)	0.97	0.97	0.97	0.94
RF(50 trees)	0.97	0.97	0.97	0.94
DT	0.95	0.95	0.95	0.90

IV. CLASSIFIER ANALYSIS

Now that the classifier was selected, according to the project plan, the next step was to use passive DNS traffic data, from where first N queries of non-existent domain names per IP address will be selected, processed and passed to the classifier to detect whether the domain names were algorithmically generated or not.

A. Data set

Appropriate labelled data sets for this purpose were unfortunately not found. The DNS traffic data sets that were available, all failed to meet one or more of the following criteria which were crucial for performing the analysis:

- The data set must include traffic generated by malwares that use DGAs
- The data set should have labels identifying which of the domain names were algorithmically generated
- The data set should include traffic data of more than one host

Most of the papers studied for the purpose of learning about data set gathering had mentioned using proprietary data sets (they mostly generated the traffic in controlled environments to collect data) which were not available publicly [6], [12] and [13].

Only one data set was found from [11] which met two of the above mentioned criteria. It included algorithmically generated domain names and there were traffic data of 22 hosts, out of which 10 were infected with Conficker malware and one demonstrated suspicious behaviour. Unfortunately, however, the data set had no labels mentioning which of the domain names were generated algorithmically, which is crucial because an infected host does not always generate algorithmically generated domain names as the host user may be using it to perform their own activities on the internet that could result in generating non-existent domain names by mistake.

B. Data Preprocessing and Feature Extraction

The collected data set included the time the traffics were generated, the source and destination IP addresses, the domain name queries and responses and more. It also included non-DNS traffic information. This data set was then processed to get our desired data set for testing through the following steps:

- Only the DNS responses mentioning non-existent domain names were extracted
- The time, source IP and the domain names were extracted
- The lexical features required by our classifier were then extracted from the domain names
- Duplicate domain names were removed
- The data set was then passed to the classifier.

C. Results

Since there were no labels available in the data set, it was not possible to determine the accuracy or confusion matrix of this test. However, after the classification was done, the unique IP addresses of the domain names which were labelled as DGA generated by the classifier were obtained.

The classifier was able to detect DGA generated domains from all the 10 hosts that were known to be infected. There was no false positives in this case, that is, the classifier did not identify any IP as infected which actually wasn't infected. It is to be noted here that while training and testing the classifier, the domain name samples used were from completely different malwares. Yet the classifier was able to detect algorithmically generated domain names of a malware that was completely new to it.

V. FURTHER EXPLORATION & EXPECTED CONTRIBUTION

The aim of real time bot net detection from DNS query data is to detect whether a host sending a non-existent domain name query is a bot or not the moment as soon as possible, preferably as soon as it receives even one of such queries. However, certain other factors, as discussed in section I need to be taken into account. Although the planned system of this project depends on several non-existent domain queries to give verdict on the arriving non-existent domain, while building the classifier, a great amount of attention was paid to ensure that the classifier was really good at classifying among the data sets. If the right kind of data set and more time were available to conduct this project, then perhaps more improvements could have been brought to the planned system. Some aspectss of future exploration are briefly discussed in the following sub-sections.

A. Prediction Algorithm

Since the appropriate data set was not available, it was not possible to implement the whole plan and test it. Hence, proper data set has to be acquired and the algorithm shown in figure 4 should be followed to experiment and come up with appropriate values of N and x.

If the algorithm test results prove to be successful, then the issue with the taking advantage of time window will be solved. To further improve the efficiency of this algorithm, the value of N can be randomized within a certain limit for each IP. This way, it will be difficult to evade detection for the bots.

B. Classifier Feature Selection

Most DGAs use a function of time to generate domain names [14] [15]. There might also be other factors that are taken into consideration by botmasters when

```
1. INITIALIZATION:
2. Q = {}; //list of IPs with their corresponding non-existent domains, initially empty
3. N = <number of IPs to be considered>;
4. x = <threshold of number of detected DGA domains>;
5. while(true)
6.     if (non-existent domain query)
7.         add IP i and domain name d pair to Q
8.         if (instances of i == N)
9.             for (all N domain d of the IP i)
10.                data = extract features
11.                dga = classify(data) //number of DGA domains detected
12.            if (dga == x)
13.                return "IP i is Bot"
14.            else
15.                remove all instance of IP i from Q
16.                return "IP i is not Bot"
```

Fig. 4. Algorithm for the system

designing their DGAs. Hence, apart from individual purely lexical features of the domain names, there might be more information hidden among the domain names, such as dependence on time.

Moreover, if the domain names were generated leveraging time as a function, then there is a high possibility that each character within a domain name will be somehow correlated to the character before and/or after it. Furthermore, if the domain names generated by an IP over time are studied without randomizing their sequences, then it is possible to find correlations among the domain names generated by that IP.

In the classifier designed for this project, each domain name was considered individually and no dependency among the characters of a domain name were considered. Even with just that, it was possible to achieve high accuracy and precision scores. If the features of correlation among characters as well as that among domain names generated from a single IP were to be considered, it might have been possible to improve the classifier further.

VI. CONCLUSIONS

Botnets impose significant amount of threats to cyber security. What gives botnets so much power is the sheer size of it. Hence, it has become very important to curb the growth of botnets to ensure safety of individuals as well as organizations on the internet. This project was a small attempt at contributing to the existing fight against botnets and their masters.

While studying the previous research works related to botnet detection, a lot of information about the characteristics and behavior of botnets were gathered. A lot of information about how botnets use DGAs, how these DGAs generate domain names and how these domain names are different from the benign domain names were collected.

After summarizing and analyzing the different aspects of botnet detection, it was decided that the focus of the project would be to detect bots that use DGAs to communicate with their CC servers. In this regard, the lexical features of algorithmically generated malicious domains and those of benign domains were calculated for a data set. It was also observed that some characters had very different frequencies of occurance in malicious and benign domain names. This factor was also considered while selecting features for the classifier.

Those lexical features were then used to compare different machine learning algorithms in order to detect domains generated by DGAs. Random Forest Classification algorithm was found to have done a better job among all the other tested algorithms, namely, Logistic Regression, K Nearest Neighbor, Decision Trees, Support Vector Machine and Gaussian Naive Bayes. An accuracy of almost 97% was achieved with the classifier using the testing data set.

Although the whole planned botnet detection system included some more functionalities aside from the classifier, due to lack of appropriate data sets and time, it was not possible to implement the whole of it. Moreover, several other factors, such as the correlation among characters within a domain name and that among domain names generated by the same IP were not considered. If those matters are employed into the classifier and the whole system is implemented, then it might lead to better results.

REFERENCES

[1] R. Sharifnya and M. Abadi, "A Novel Reputation System to Detect DGA-Based Botnets", in 3rd International Conference on Computer and Knowledge Engineering (ICCKE), 2013, pp. 417-423.

[2] C. Han and Y. Zhang, "CODDULM: An approach for detecting C&C domains of DGA on passive DNS traffic", in 6th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2017, pp. 385-388.

[3] M. Feily, A. Shahrestani and S. Ramadass, "A Survey of Botnet and Botnet Detection", in 2009 Third International Conference on Emerging Security Information, Systems and Technologies, Athens, Glyfada, Greece, 2009, pp. 268-273.

[4] M. Grill, I. Nikolaev, V. Valeros and M. Rehak, "Detecting DGA malware using NetFlow", in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 2015, pp. 1304-1309.

[5] Erquiaga, M., Catania, C. and Garca, S. (2016). Detecting DGA malware traffic through behavioral models. In: 2016 IEEE Biennial Congress of Argentina (ARGENCON). IEEE, pp.1-6.

[6] F. Bisio, S. Saeli, P. Lombardo, D. Bernardi, A. Perotti and D. Massa, "Real-time behavioral DGA detection through machine learning", in 2017 International Carnahan Conference on Security Technology (ICCST), Madrid, Spain, 2017, pp. 1-6.

[7] H. Mac, D. Tran, V. Tong, L. Nguyen and H. Tran, "DGA Botnet Detection Using Supervised Learning Methods", in SoICT 2017 Proceedings of the Eighth International Symposium on Information and Communication Technology, Nha Trang City, Viet Nam, 2017, pp. 211-218.

[8] X. Hoang and Q. Nguyen, "Botnet Detection Based On Machine Learning Techniques Using DNS Query Data", Future Internet, vol. 10, no. 5, p. 43, 2018.

[9] osint.bambenekconsulting.com, 2018. [Online]. Available: http://osint.bambenekconsulting.com/feeds/. [Accessed: 05- Jul- 2018].

[10] alexa.com, 2018. [Online]. Available: https://www.alexa.com/. [Accessed: 04- Jul- 2018].

[11] Stratosphere IPS, 2018. [Online]. Available: https://www.stratosphereips.org/. [Accessed: 05- Jul- 2018].

[12] S. Yadav, A. Reddy, A. Reddy and S. Ranjan, "Detecting algorithmically generated malicious domain names", in IMC '10 Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, Melbourne, Australia, 2010, pp. 48-61.

[13] T. Nguyen, T. CAO and L. Nguyen, "DGA Botnet detection using Collaborative Filtering and Density-based Clustering", in SoICT 2015 Proceedings of the Sixth International Symposium on Information and Communication Technology, Hue City, Viet Nam, 2015, pp. 203-209.

[14] S. Schiavoni, F. Maggi, L. Cavallaro and S. Zanero, "Tracking and Characterizing Botnets Using Automatically Generated Domains", 2013.

[15] A. Bonneton, D. Migault, S. Senecal and N. Kheir, "DGA Bot Detection with Time Series Decision Trees", in 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), Kyoto, Japan, 2015, pp. 42-53.

[16] T. Wang and L. Chen, "Detecting Algorithmically Generated Domains Using Data Visualization and N - Grams Methods", in Proceedings of Student - Faculty Research Day, CSIS, Pace University, 2017.

[17] A. Ahluwalia, I. Traore, K. Ganame, N. Agarwal. (2017) Detecting Broad Length Algorithmically Generated Domains. In: Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017.