This is the Camera-ready version of the paper published in IEEE MILCOM, Norfolk, VA, USA, Nov 12-14, 2019

Domain Flux-based DGA Botnet Detection Using Feedforward Neural Network

Md. Ishtiaq Ashiq[1], Protick Bhowmick[1], Md. Shohrab Hossain[1], Husnu S. Narman[2]
[1]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh
[2]Weisberg Division of Computer Science, Marshall University, Huntington, WV, USA
Email: ishtiaqashiq5@gmail.com, protickbhowmick1994@gmail.com, mshohrabhossain@cse.buet.ac.bd, narman@marshall.edu

***Abstract*—Botnets have been a major area of concern in the field of cybersecurity. There have been a lot of research works for detection of botnets. However, everyday cybercriminals are coming up with new ideas to counter the well-known detection methods. One such popular method is domain flux-based botnets in which a large number of domain names are produced using domain generation algorithm. In this paper, we have proposed a robust way of detecting DGA-based botnets using few novel features covering both syntactic and semantic viewpoints. We have used Area under ROC curve as our performance metric since it provides comprehensive information about the performance of binary classifiers at various thresholds. Results show that our approach performs significantly better than the baseline approach. Our proposed method can help in detecting established DGA bots (equipped with extensive features) as well as prospective advanced DGA bots imitating real-world domain names.**

***Index terms*—** Botnet detection, Domain Flux, DGA, Neural Network, C&C Server, HMM, edit distance, ROC curve

I. Introduction

Military communication involves the transmission of heavily secured information. These operations entail communication between personnel not only within the same nation but also with security organizations internationally. Thus, even a minor infiltration of military network can be catastrophic in context of the national risk it poses as well as the potential consequences it will have internationally. Now one way of invading into this network is botnet. Using botnet, adversaries can collect a vast amount of private information, commit financial fraud, identity theft, launch a DDoS attack or distribute other malwares, such as viruses or adwares [1]. That is why, in this paper, we have proposed a methodology that will help to preclude DGA botnet attacks in military network and reveal potential security breaches upfront.

Among different types of botnets, one notorious variant uses of domain fluxing method, in which botmaster constantly changes the domain name of the Command and Control (C&C) server very frequently. These domains are produced using an algorithm called Domain Generation Algorithm (DGA). Domain flux-based botnets are stealthier and consequently much harder to detect due to its flexibility.

There have been few research works [2]–[10] on DGA-based botnet detection. Yadav et al. [2] used a methodology based on the observation that DGAs do not generally produce well-formed and pronounceable domain names. This implies that algorithmically generated domain names will exhibit vastly different characteristics from legitimate domain names. Raghuram et al. [3] proposed a method for detecting anomalous domain names with a probability model. Zhang et al. [4] tried to detect DGA domain names by extracting meaningful morphological units from domain names on the basis that the distribution of morphemes would be different in human-generated domains and DGAs. Mowbray et al. [6] worked on detecting DGA domain names by inspecting DNS queries. They examined second-level string lengths in the domain names to find unusual lengths that differ greatly from benign domain names. Marchal et al. [7] proposed a method that detects malicious domain names by comparing its semantic similarity with known domain names. Han et al. [9] proposed a method to identify DGA domains by lexical attributes and then using Support Vector Machine as classifier. Due to the vulnerability of static feature based methodology, another recent work [10] suggested a combination of the N-gram model with deep convolutional neural network to classify between DGA and benign domains. Â

The basic limitations of some of the aforementioned works [3], [6], [7] are that if random meaningful word phrases are coupled together, they might not work. In [4], repetition of words and suffixes in domain names back and forth will neutralize the algorithm. DGA domains showing a bit of regularity will fail [6].

Our work *differs* from the previous works that we have exploited the irregularities in DGA domains from both lexical and contextual standpoint. We have tried to find out the syntactic mismatches as well as the semantic inconsistencies in the DGA generated domains.

The *objective* of this paper is to find out a robust way to detect the presence of DGA domains with better accuracy. The *contributions* of this work are: (i) we have developed a heuristic for evaluation and detection of botnets inspecting the several attributes in a very simple and efficient way, (ii) We have compared our proposed system with the existing ones with respect to accuracy. Our proposed method can help in detecting established DGA bots (equipped with extensive features) as well as prospective advanced DGA bots imitating real-world domain names.

The rest of the paper is organized as follows. In Section II, we propose our detection methodology. Details about the dataset and performance metrics are described in Section III. Results of our approach are presented in Section IV. Finally, Section V has the concluding remarks.

II. Proposed Approach

We model a feed-forward Neural Network in order to partition DGA generated domain names from benign domain names. We have used eight robust features: five of them represents the syntactic congruity whereas the rest three represents the semantic appropriacy of domain names.

A. Features

We have chosen the following features to build our classifier:

1) Length: Early DGAs generally show uniformity in the length of domain names. Domain length might be inadequate in the detection of advanced DGA, but is a good indicator of malignancy in many cases. For example, the Zeus bot domain contains domains like "ceqolfhbqcuwqkxbmayh" or "lbydhalbnzpwcpfbytuqwdatcor". The abnormality in length of these domains suggests that they are not human given domain names.

2) Vowel-Consonant Ratio: This feature is a good indicator of DGA generated algorithms. DGAs make random domain names with an approximately constant ratio of vowels and consonants periodically inserting vowels in between consonant strings. Low vowel consonant ratio can indicate that the domain name might be generated with an algorithm.

3) Four-Gram Score: This feature is a score value depending on the number of contiguous substrings without a vowel, which will be very advantageous in case of DGA domains with reasonable vowel-consonant ratio. For example, "tdtxaaaa" might be a good candidate of DGA generated domain name. However, with a vowel-consonant ratio of 0.5, it is similar to real-world words or phrases. This feature takes into account the order of the letters. A long string of consonants (or vowels) is uncommon in a language, making it unpronounceable.

For the calculation of this score, we find out the number of substrings of length four without a vowel within a domain name, and the ratio of that number to the length of the domain name is our score. We have chosen substring length as four because generally, four consecutive letters in a pronounceable word do have a vowel in between.

4) Regularity Score: The regularity score takes into account the syntactic dissimilarity with actual words. Here we use the edit distance as our similarity measure. Edit distance takes two words as function parameters and returns the minimum number of deletions, insertions, or replacements to transform one word into another. However, as we want to find the closest matching words for a certain domain from possibly many thousands of words, this approach would be too inefficient. To make it efficient, we have used trie data structure where all shared prefixes of all the words will follow the same path. Therefore, there is no need to search through the whole dictionary to find our closest matching. This whole process is described in further detail in [11].

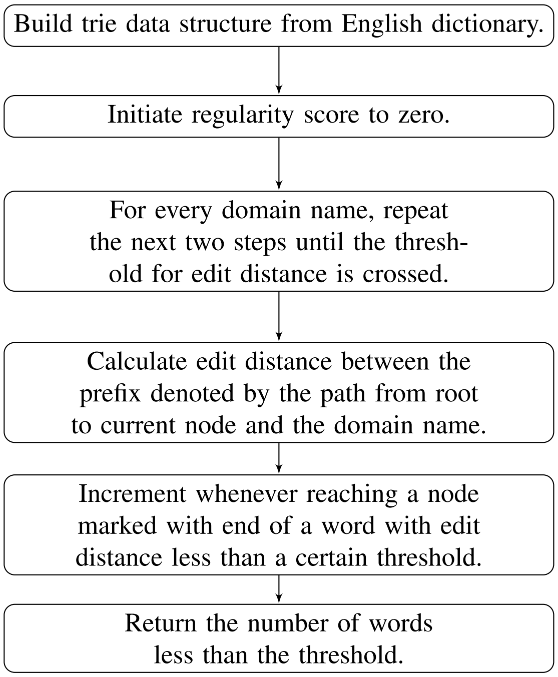

Build trie data structure from English dictionary.
Initiate regularity score to zero.
For every domain name, repeat the next two steps until the threshold for edit distance is crossed.
Calculate edit distance between the prefix denoted by the path from root to current node and the domain name.
Increment whenever reaching a node marked with end of a word with edit distance less than a certain threshold.
Return the number of words less than the threshold.

Fig. 1: Regularity Score step by step

The maximum cost is based on the length of the domain. We use the number of bits required to represent the length of the domain to denote the maximum cost (c) and search for words in the dictionary that are c edits away from our domain. This word count is used as our feature value. The whole procedure is illustrated in Fig. 1 flow chart.

5) Markov Score: This score is intended for finding out the randomness of the domain name. The more random the letter distribution within a domain name is, the more likely it is to be a DGA generated domain name. We have used a basic Markov model to find the probability of a transition within a 2-gram. Markov score is a good indicator of the randomness of a domain name, taking into account how likely the one step transitions are within it.

6) Meaning Score: Real world domain names tend to include meaningful words or phrases (there are exceptions, but the majority shows this trend). Most of the widespread domain generation algorithms use pseudo-random number generator for creating the domain names, and the resulting domain names have a negligible number of meaningful sub-phrases within. This feature can be exceptionally useful for DGAs that do not affix real words or expressions to form domain names.

For the calculation of this score, we extract the number of meaningful substrings or phrases within the domain name and generate a meaning score from that number by normalizing with respect to the length of the domain name.

7) Frequency Score: This score denotes the frequency of words and phrases within the domain names used over the Internet. It will work effectively for detecting DGAs that concatenates random phrases from the dictionary. The fact that domain names are more likely to contain widely used words contributes to the strength of this feature. We have used an open source API *phrasefinder* to get the frequency scores from Google corpus. Each sub strings of length greater than three are candidates for the score and they have an exponentially weighted contribution according to their lengths. Non-proportionate scaling is used as frequency of sub-strings over Internet has a skewed decrease with respect to length.

8) Correlation Score: We have gone through the relevance of a domain name from the perspective of meaning, frequency of use and randomness. However, our scores were all from segregated segments of a domain name, none of the features worked on the correlation of those segments. In simple words, two substrings within a domain name might pass every previous checkpoint, but there might be no correlation between them, which will make the validity of the domain name null. The calculation procedure is illustrated in Algorithm 1.

Symbols Used: Correlation Matrix (CM), Reference Text File (r), correlation score (cs), Correlation Matrix entry : occurrence of word pair a and b in same sentence (cm_a^b), input domain name (d)

Algorithm 1: Correlation Score step by step

```
procedure CORRELATION SCORE
    for $l \in$ lines in $b$:
        for every pair of consecutive words $a, b \in l$:
            $cm_a^b$++
    Normalize CM
    $cs \leftarrow 0$
    for each consecutive sub-string pair $a, b \in d$:
        $cs \leftarrow cs + cm_a^b$
    return $cs$
```

B. Justification of robustness

The whole idea of our features comes from the fact that human given domain names are pronounceable, sometimes, contextually meaningful, does not contain improbable letter combination and generally are combinations of two or more meaningful words. We have derived our features from these attributes. Our four-gram score will detect domains that will pass length and vowel-consonant ratio but will be highly random, such as "tdattxa". Markov score is able to detect improbable letter combination. Meaning score and frequency score are there to detect syntactically regular but meaningless domains such as "eftanne". Regularity score tries to match how syntactically close a given domain name is to the actual dictionary words. Humane domain names (even misnomer ones) will exhibit regular words. Correlation score is able to detect domains that are generated by smart bots. This score detects the contextual similarity of word parts within the domain name.

III. Experiments

In this section, we present a detailed discussion of our collected dataset, our performance metric, and hyperparameters used in our experiment.

A. Dataset Details

We have collected our dataset from [12]. They used two different algorithms to produce two sets of data with different metrics. Two algorithms were based on the Hidden Markov Model and Probabilistic Context Free Grammar. Domains generated from these two DGAs are much harder to detect [12] and have been used for the evaluation of our proposal. It illustrates flaws in the lexical feature based DGA detection system such as [2]. Another repository of the dataset contained a set of known botnets and their used domain names. We explain briefly these two datasets in the following two paragraphs.

In HMM-based dataset, eight files (DNL1, DNL2, DNL3, DNL4, 9ML1, 500KL1, 500KL2, 500KL3) were generated varying two parameters. One is L - maximum number of history symbols used to generate the HMM transition probability, and other is the dictionary for constructing the model. DNL files were built with English dictionary as the source. 9ML1 file was generated from 9 million domain names from IPv4 space, and 500KL files were generated from randomly selected 500000 entries from this 9 million IPv4 space. Words of last four files do have numbers in between and are less likely to be pronounceable whereas the probability of having pronounceable words in DNL files, especially in DNL3 or DNL4, are quite high.

In PCFG-based dataset, four files (pcfg_dict, pcfg_dict_num, pcfg_ipv4, pcfg_ipv4_num) were generated using the English dictionary with hyphenation along with a list of IPv4 domain names. Here, each word was derived from a set of production rules in a Grammar $G(N, \sum, R, S)$. Each rule also contained a probability associated with it. First, two files were generated using English syllables as the source and as such contained pronounceable words. Last two were generated using IPv4 syllables. There is another directory in the dataset `other` that contains a list of domain names generated from some known Botnets such as *Zeus*, *Conflicker*, *Kraken*, *Srizbi*, *Torpig*.

It contains another file named *Kwyjibo* which contains some very pronounceable and short English words such as 'slitingly' or 'underlining'. Domains of this file will not only work as a metric to judge how good our system does to detect pronounceable short DGAs but also will give us an idea about the number of false positives in our system. Too good a performance in this file may indicate a high false positive rate in DGA domain detection because these words may not have an associated legitimate domain now, but they may be registered as one in future.

TABLE I: Symbols and their meaning

Name	Symbol	Name	Symbol
True Positive	TP	True Negative	TN
False Positive	FP	False Negative	FN
Accuracy	α	F1_score	f
Precision	p	Recall	r
Learning Rate	ϵ		

B. Performance Metric

We have used the following performance metrics for evaluation of our system. Symbols used are listed in Table I.

1) Accuracy: Accuracy of botnet classification is denoted by the percentage of correctly predicted label in test data, calculated as follows:

$$\alpha = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2) F1 Score: F1 score can be comprehended as a weighted function of recall and precision, computed as follows:

$$f = 2 * \frac{p * r}{p + r} \tag{2}$$

Here, Precision (p) is the ratio of correctly predicted positive results to all positive predictions by the classifier. Recall (r) is the ratio of correctly predicted positive results to all the samples that should be predicted as positive.

3) ROC curve and AUC score: The full form of AUROC is the area under the Receiver Operating Characteristic curve. It can be interpreted as the percentage of randomly drawn samples that were given correct labels. To compute the AUROC value, we need to build Receiver Operating Characteristic (ROC) curve first. Hence, FP rate (FPR) and TP rate (TPR) are calculated for different thresholds, and they are plotted in a single graph, where TPR values are on the y-axis, and FPR values are plotted along the x-axis. Equation of FPR and TPR is the following:

$$FPR = \frac{FP}{TN + FP} \tag{3}$$

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

The resulting curve is called ROC curve. The AUROC value or AUC score is the area under this curve. In an ideal case, ROC curve will touch the leftmost $FPR = 0$ line and then follow the $TPR = 1$ line, and the AUC score will be 1. Consequently, we can visually judge the performance of our classifier looking at the distance from $y = x$ line.

C. Experimented Values

We have trained our feedforward neural network model with the feature values generated from our dataset. Our baseline approach is [2]. In order to train our neural network, we varied our hyperparameters according to Table II. After running several times, we got an approximation that Learning rate (ϵ) 0.05, epoch 150, four hidden layers with 40, 30, 20 and 20 neurons in each layer respectively provided the best result. Our training accuracy is slightly better for four hidden layers than 2 or 3. One thing to note here is that for easily detectable domains, less hidden layers produce slightly better results, however, for files with hard-to-detect domain names, use of 4 hidden layers gives considerably better output. Hence, the above parameters perform better on average. Here, we limit our hidden layers to 4 to prevent overfitting.

We used Adam optimizer in our code. Initial epochs may contain few spikes indicating that the optimizer is yet to find the correct path to a minimum point in the cost function vs. input graph. As stated in the previous section, our cost function is mean squared error.

TABLE II: Hyperparameters used during training

Hyper parameter	Range
Learning Rate (ϵ)	0.05-0.6
Epoch	120-200
Hidden Layer	2-4
Cross Fold	No fold, 3, 5
Neurons in hidden layers	10-50, 7-40, 5-30, 2-20
Number of runs	200

TABLE III: Results

File Name	Test Accuracy	F1 score	AUC score	Comment
9ML1	92%	0.92	0.96	Excellent
500KL1	96%	0.96	0.98	Excellent
500KL2	95%	0.95	0.98	Excellent
500KL3	86%	0.86	0.93	Excellent
DNL1	85%	0.85	0.92	Excellent
DNL2	82%	0.82	0.89	Excellent
DNL3	81%	0.81	0.88	Good
DNL4	81%	0.81	0.88	Good
kraken	96%	0.96	0.99	Excellent
srizbi	97%	0.97	0.98	Excellent
torpig	99%	0.99	0.99	Excellent
zeus	100%	1.00	1.00	Excellent
conflicker	97%	0.97	0.98	Excellent
kwyjibo	70%	0.70	0.79	Moderate
pcfg_dict	70%	0.70	0.77	Moderate
pcfg_dict_num	73%	0.73	0.79	Moderate
pcfg_ipv4	85%	0.85	0.92	Excellent
pcfg_ipv4_num	86%	0.86	0.94	Excellent

IV. Results

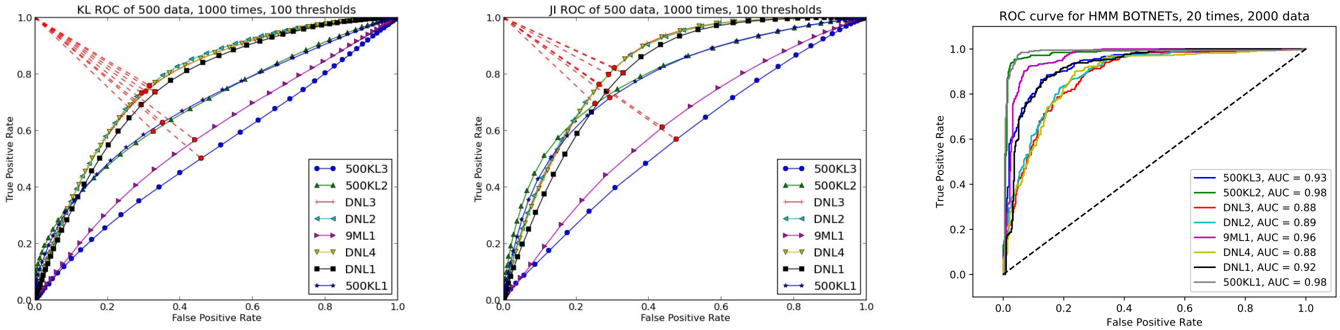
We have compared our findings with [2] which used a methodology based on the observation that DGAs do not generally produce well-formed and pronounceable domain names. They proposed three different metrics to calculate the probability of a domain belonging to a particular botnet: Kullback-Leibler distance, Edit Distance, and Jaccard Distance [2]. These detection schemes yielded up to 100% accuracy with a very low false positive rate. Hence, we have compared our findings with the result of this approach.

The test phase was done on the test data (20% of total samples). The result is shown in Table III. If AUC score is greater than 0.9, we call it $excellent$. If it falls within the range 0.80-0.9, it is *good*. Within 0.70-0.80 is $moderate$ and anything less than 0.70 is termed as $poor$. These results are the approximate average of running the code 1000 times.

A. Comparing HMM-based DGAs

Fig 2 shows the comparisons between our results and the results achieved from the two methods described in [2]. The first two pictures were collected from [12]. Here, we see ROC curve for the files in our 'hmm_dga' directory.

As described earlier, 'DN' files were produced from English dictionary using HMM. So, the letter sequence followed a real-world pattern which made some of our features (length, vowel-consonant ratio, four-gram score, or Markov score) ineffective. Hence, ROC curve for highly random domains in 9ML1 or 500KL1 files encompasses the upper region of $y = x$ and

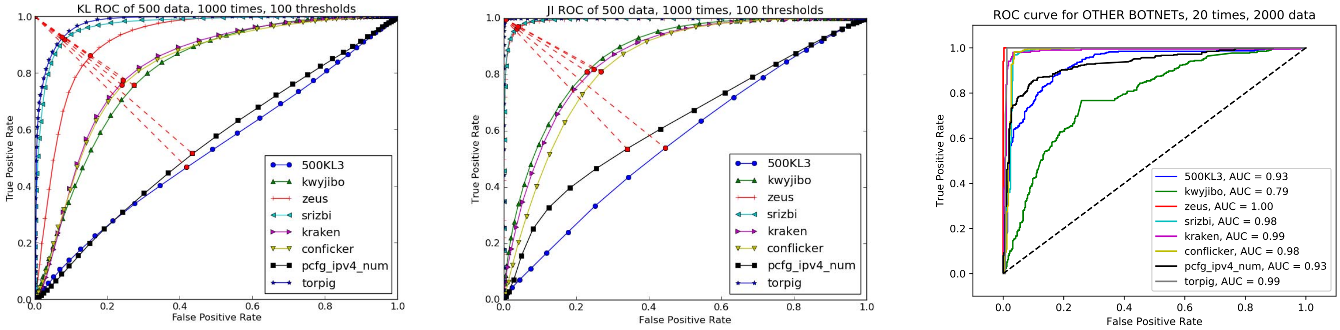


(a) using KL distance [12]

(b) using Jaccard Index [12]

(c) Our approach

Fig. 2: Comparison between baseline approach and our approach for 'hmm' dataset

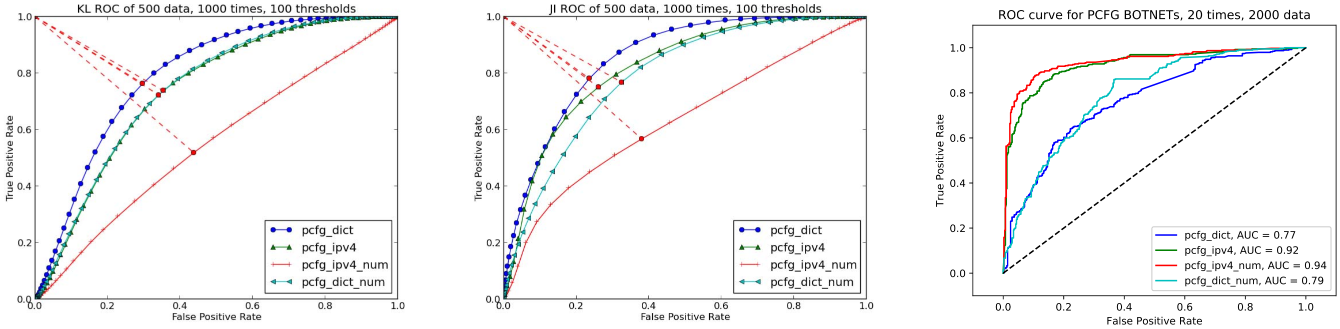


(a) using KL distance [12]

(b) using Jaccard Index [12]

(c) Our approach

Fig. 3: Comparison between baseline approach and our approach for 'other' Dataset



(a) using KL distance [12]

(b) using Jaccard Index [12]

(c) Our approach

Fig. 4: Comparison between baseline approach and our approach for 'pcfg' dataset

area under the curve is high but curve for DNL3, or DNL4 is comparatively closer to $y = x$ line.

B. Comparing Real DGAs

Fig 3 shows the comparisons between our results and the results achieved for the files in our 'other' directory. Results for this directory is very satisfactory. Conflicker, Zeus, Srizbi, Torpig, and Kraken contain highly random words. So, our syntactic features are adequate for detection of these domains. Results for Kwyjibo is slightly poorer because it contains domain like 'ending', 'devouringly' or 'underlandings' which depict real-world words. These domains are not only pronounceable but also realistic. Even if they may not exist now, but definitely there could be domains like *www.ending.com* in the future. Thus, in a way, our system is better because the number of false positive in our system will be low. For the ones that got detected even after being pronounceable is due to our correlation and frequency score. Correlation score will point out the incoherence between word segments of the domain name. For example, 'sulfateful' contains two very improbable word segments - 'sulfate', a chemical compound and 'ful', a vastly used English word suffix. Obviously, the correlation score would be low in this case. Frequency score will depict relatively lower usage over the internet.

C. Comparing PCFG based DGAs

Fig 4 shows the comparisons between our results and the results achieved for the files in 'pcfg directory. As these domains were produced from English syllables, they are highly pronounceable. For the same reasoning as 'Kwyjibo'

TABLE IV: Quantitative comparison with Baseline Approach

File	KL score	JI score	Our result
9ML1	0.70	0.70	0.96
500KL1	0.84	0.90	0.98
500KL2	0.86	0.92	0.98
500KL3	0.55	0.62	0.93
DNL1	0.84	0.91	0.92
DNL2	0.83	0.90	0.89
DNL3	0.84	0.89	0.88
DNL4	0.84	0.87	0.88
kraken	0.88	0.90	0.99
srizbi	0.95	0.91	0.98
torpig	0.95	0.99	0.99
zeus	0.94	1.00	1.00
conflicker	0.89	0.88	0.98
kwyjibo	0.81	0.89	0.79
pcfg_dict	0.80	0.88	0.77
pcfg_dict_num	0.75	0.86	0.79
pcfg_ipv4	0.75	0.88	0.92
pcfg_ipv4_num	0.58	0.60	0.94

domains, results are slightly inferior for ‘pcfg_dict’ and ‘pcfg_dict_num’ files. ‘pcfg_ipv4_num’ and ‘pcfg_ipv4’ has a better score because our correlation score was able to detect the incoherence among domain word segments and corpus score indicated the low or no use of these domains over the internet. Another important thing to note, ‘pcfg_dict_num’ and ‘pcfg_ipv4_num’ contain numbers. So results should be theoretically better compared to dict or ipv4 files as the regularity score will come into account but AUC score is almost same for ’pcfg_dict’ and ‘pcfg_dict_num’ because of their highly realistic nature.

D. Comparison with baseline approach

Table IV shows a comparison with our baseline approach. The classifier of [2] could not classify ‘500KL3’ (which is the best HMM-based DGA [12]) properly, and it had an AUC score of around 0.5. Our neural net has been able to detect this DGA quite well. ‘pcfg_ipv4_num’ is the best PCFG based DGA and again our system had a significantly high score than [2]. Key findings from this table can be categorized into two groups. For files consisting of words with highly random and improbable letter combination, our score is excellent. Our syntactic features are adequate enough to separate the domains into two groups in this case. For files consisting of words with highly pronounceable nature (specially without any numbers in between), our score is slightly inferior than other files yet better than our baseline approach. In this case, the correlation score or frequency score are able to find the incoherence among word segments or absence of the word segments over the corpus dataset.

In Fig. 5, we have shown a comparison using bar graph with baseline approach. It shows that on average, our result is significantly better than the two scores of the baseline approach [2]. More importantly, the confidence interval suggests that variation of result in our system are not be as much as the other two methods.

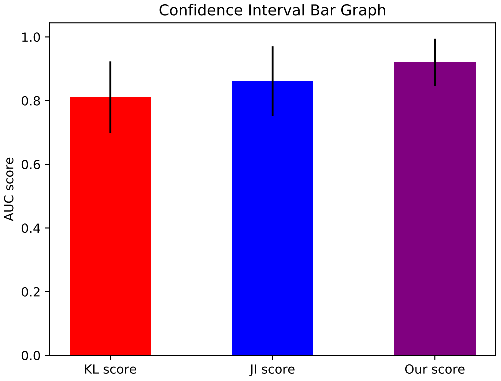


Fig. 5: Confidence Interval Bar Graph

V. Conclusion

In this paper, we proposed a robust DGA domain detection system using the feedforward neural network. Our proposed diverse features extensively tackle the problem from two aspects: syntactically and semantically. The result is exceptionally well on DGAs that use pseudo-random number generator. Our regularity score and Markov score exploits this trait for better detection. For DGAs that use pronounceable domain names, frequency score, and meaning score are useful for evaluating the validity of the phrases within the domain name. When related phrases and words appear within the domain names, the correlation score represents the association and correlation of them. Overall results show that our method can detect DGA botnets with commendable accuracy.

References

[1] B. Hammi, S. Zeadally, and R. Khatoun, “An empirical investigation of botnet as a service for cyberattacks,” *Transactions on Emerging Telecommunications Technologies*, vol. 30, pp. 1068–1082, 2019.

[2] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, “Detecting algorithmically generated domain-flux attacks with DNS traffic analysis,” *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, 2012.

[3] J. Raghuram, D. J. Miller, and G. Kesidis, “Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling,” *Journal of Advanced Research*, vol. 5, no. 4, pp. 423–433, 2014.

[4] W.-w. Zhang, J. Gong, and Q. Liu, “Detecting machine generated domain names based on morpheme features,” in *1st International Workshop on Cloud Computing and Information Security (IEEE CloudCom)*, Bristol, UK, Dec 2-5, 2013.

[5] M. S. Hossain, A. Paul, M. H. Islam, and M. Atiquzzaman, “Survey of the protection mechanisms to the SSL-based session hijacking attacks,” *Journal of Network Protocols and Algorithms*, vol. 10, no. 1, pp. 83–108, 2018.

[6] M. Mowbray and J. Hagen, “Finding domain-generation algorithms by looking at length distribution,” in *IEEE International Symposium on Software Reliability Engineering Workshops*, Naples, Italy, Nov 2014.

[7] S. Marchai, J. Francois, R. State, and T. Engel, “Semantic based dns forensics,” in *International Workshop on Information Forensics and Security*, Tenerife, Spain, Dec 2012, pp. 91–96.

[8] H. S. Narman, M. S. Hossain, and M. Atiquzzaman, “Multi class traffic analysis of single and multi-band queuing system,” in *IEEE GLOBECOM*, Atlanta, GA, USA, Dec 9-13, 2013.

[9] C. Han and Y. Zhang, “CODDULM: an approach for detecting C&C domains of DGA on passive DNS traffic,” in *International Conference on Computer Science and Network Technology*, Dalian, China, Oct 2017.

[10] C. Xu, J. Shen, and X. Du, “Detection method of domain names generated by DGAs based on semantic representation and deep neural network,” *Elsevier Computers and Security*, vol. 85, pp. 77–88, August.

[11] “Fast and Easy Levenshtein distance using a Trie,” http://stevehanov.ca/blog/index.php?id=114.

[12] F. Yu, L. Yu, O. Hambolu, and et al., “Stealthy Domain Generation Algorithms,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 1430–1443, 2017.