MaldomDetector: A System for Detecting Algorithmically Generated Domain Names with Machine Learning

Ahmad O. Almashhdani[1,2], Mustafa Kaiiali[3], *Senior Member, IEEE,* Domhnall Carlin[1], Sakir Sezer[1]
[1] Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, UK
[2] Computer Center, University of Baghdad, Baghdad, Iraq
[3] Cyber Technology Institute, De Montfort University, Leicester, UK

Abstract

One of the leading problems in cyber security at present is the unceasing emergence of sophisticated attacks, such as botnets and ransomware, that rely heavily on Command and Control (C&C) channels to conduct their malicious activities remotely. To avoid channel detection, attackers constantly try to create different covert communication techniques. One such technique is Domain Generation Algorithm (DGA), which allows malware to generate numerous domain names until it finds its corresponding C&C server. It is highly resilient to detection systems and reverse engineering, while allowing the C&C server to have several redundant domain names. This paper presents a malicious domain name detection system, MaldomDetector, which is based on machine learning. It is capable of detecting DGA-based communications and circumventing the attack before it makes any successful connection with the C&C server, using only domain name's characters. MaldomDetector uses a set of easy-to-compute and language-independent features in addition to a deterministic algorithm to detect malicious domains. The experimental results demonstrate that MaldomDetector can operate efficiently as a first alarm to detect DGA-based domains of malware families while maintaining high detection accuracy.

Keywords: Network Security, Intrusion Detection, Machine Learning, Command and Control, Domain Generation Algorithm (DGA), DNS, Domain name.

1. Introduction

Cybercriminals often depend on command and control (C&C) to launch their cyber-attacks. The cybercriminals can run many malicious activities, such as data exfiltration, spamming and downloading harmful files by directing the compromised machines via C&C channels [1]. To protect these channels from being detected and blocked by security controls, attackers have started to build reliable C&C infrastructures to conduct their cyber-attacks remotely. Therefore, different C&C architectures and communication techniques have emerged to make any channel detection and disruption process more difficult [2].

One of the main C&C communication techniques is *domain fluxing*, where numerous domain names are generated algorithmically by a bot or malware using a Domain Generation Algorithm (DGA). The malware then attempts to contact its C&C server by querying these domain names one after another, until an active one is found/resolved [3] [4]. DGA is a deterministic algorithm that can be used to generate many arbitrary domains, so that the attacker must only register one or more of them in advance to enable the malware to contact its C&C [5]. The DGA approach is a recent development in malware communications and has several advantages. It makes the process of retrieving C&C information, i.e., pseudo-random domains, from the malware code using reverse engineering methods very difficult [6]. This issue is only increased considering the large number of new malware families and their variants detected every day [7]. It also makes the task of identifying and locating the malicious domain names difficult for law enforcement agencies, due to the generation of many pseudo-random domain names [8]. Additionally, DGA provides a large amount of redundancy in the C&C server, where if one server is taken down, a new one can be available within a short time [2].

Advanced types of malware such as botnets and ransomware have utilized the DGA technique to make a connection with their C&C server, using DNS queries to launch a range of harmful actions [9] [10] [11]. Therefore, detecting and blocking the DGA-based domain names will disrupt the secret channels

between the victims and the controllers, i.e., cybercriminals, and thus limit the harmful effects of these sophisticated malware attacks [12].

Domains generated by DGA do not usually employ typical word-based domains, however, identifying them is a difficult process [13]. In this paper, a detection system, MaldomDetector, has been proposed to detect DGA-based domain names effectively, before any successful connection with the C&C server can be made. It consists of two modules. Firstly, the Data Preparation Module, which extracts informative features employing a deterministic algorithm called Randomness Measuring Algorithm (RMA), to measure the randomness in the domain name characters. Secondly, the Decision Making Module, which runs a machine learning classifier to process the extracted features and produce a decision offering high detection accuracy.

The remained of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes MaldomDetector's high-level architecture. Section 4 discusses MaldomDetector's implementation, including feature extraction, dataset creation, and building and evaluating the classifier. Section 5 discusses the properties of the DGA-based domain names detection systems. Finally, Section 6 concludes the paper.

2. Related Work

Several research works have attempted to detect algorithmically generated domain names through analysing the names' strings. Xu et al. [14] proposed a method to detect DGA-based domain names through inspecting the domain's character strings, by combining an n-gram approach and deep convolutional neural network. They provided evaluation results for different types of DGA using bigram and trigram representations. The average accuracies for 2-gram and 3-gram were 94.15% and 98.29% respectively. However, n-gram based approaches are computationally expensive and language-dependent.

YU et al. [15] Investigated the use of deep neural network techniques to detect DGA-based domains based solely on the domain name characters, trained with large amounts of heuristically labelled real traffic. However, this method has many parameters that must be estimated, causing high computational cost, and requires large amounts of training data to learn, i.e., has long training times. Selvi et al. [16] presented a machine learning approach ,which used a Random Forest algorithm to detect DGA-based domain names. They extracted features that rely on several characteristics, such as the lexical attributes of the domain names, some statistical information, and masked N-grams. The experimental results showed a detection accuracy of 98.91% and false positive rating of 0.76%. However, most of the extracted features are statistical (i.e. mean, variance, and standard deviation) that become less effective when the domain name is short. Additionally, this method requires a relatively high training time (approximately 1.21 hours).

Lv et al. [17] analysed malicious and benign domain names using a Hidden Markov Model (HMM), where a total of 12 attributes from five categories were extracted from the characteristics of DNS communications. One of these categories relates to the characters of the domain name. HMM requires expensive computations and has a time due to the dependence on a set of features extracted from the DNS response packet. The classification accuracy and the recall rate obtained from the test results were 91.52% and 89.32% respectively. Mac et al. [18] presented a thorough investigation on various methods, such as supervised learning techniques, Hidden Markov Model (HMM), and bidirectional Long Short-Term Memory Network (LSTM) to detect botnet attacks based on DGA. A detailed comparative analysis of these methods was presented. The maximum precision and recall obtained from this research were 92.32% and 93.09% respectively. However, just like n-gram-based approaches, Markov-based techniques are language dependent and require expensive computations.

Shi et al. [19] proposed the Extreme Learning Machine (ELM) technique to detect harmful domain names. Three out of the nine features were lexical attributes. However, the rest of the features require data from the DNS responses and access to the information in the WHOIS lookup web service, both of which may increase time taken to identify the malicious domains. The used dataset was imbalanced, and

the accuracy rate was 96.28%. Song et al. [20] presented a method based on a Random Forest classifier to detect algorithmically generated domain names. Ten features were extracted from the characters of the domain name. Some features were based on n-gram computations. The achieved precision was 93.5% and the false positive rate was 3.49%. Truong and Cheng [21] proposed a system for detecting domain fluxing by analysing DNS traffic and extracting lexical features from the character strings in malicious and benign domain names. A machine learning method was used to build this system, achieving 92.3% accuracy and 4.8% false positive rate.

The methods discussed above are based on a set of features that are either extracted out of whole DNS communications (i.e., DNS query and response packets) or that require data from external sources, such as WHOIS site. Furthermore, most of them are language-dependent and require complex computations. However, building a system that can accurately identify malicious domain names based on features extracted from the initial DNS requests would be preferable as it offers potentially earlier detection. This paper proposes a machine learning-based detection system, MaldomDetector. It provides high detection accuracy (~ 98%) and low false positive rate (~ 4%) depending solely on character level features extracted out of the domain name string of the initial DNS request. These features are easy to compute and do not need to access any external sources.

3- MaldomDetector High-Level Architecture

The architecture of MaldomDetector is presented in this section. As depicted in Fig.1, it consists of two modules. The first, the Data Preparation Module, is used to preprocess the incoming DNS request packet and extract the feature vector (V) out of the characteristics of the domain name string. V is composed of two types of features: basic and derived. The basic feature set {F1, F2, F3, F4, F5, F6} is directly extracted from the requested domain name, whereas features {F7, ... , F15} are derived from the basic features. F16 is a derived feature that represents the output of RMA. RMA is a deterministic algorithm that accepts a subset of the basic features, i.e., {F1, F2, F3, F4}, as an input, and measures the randomness in the domain name characters, determining initially whether it is malicious or benign. RMA is discussed further in Section 4.3.

The second module, the Decision-Making Module, is a machine learning-based domain name classifier. It accepts the entire feature vector V [F1, …, F16] as input and classifies it as either a malicious or benign domain name.

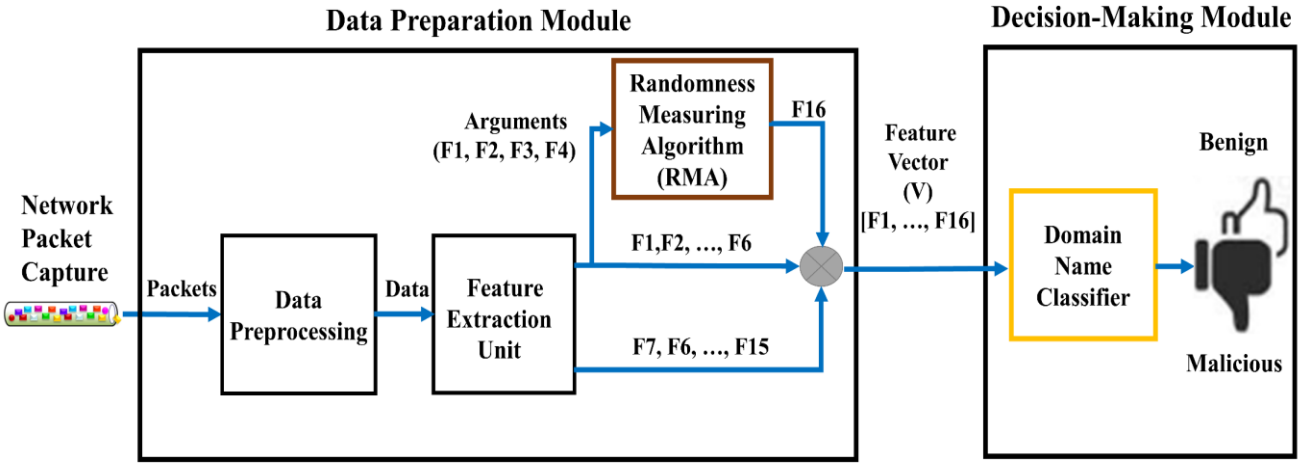


Fig. 1. MaldomDetector architecture

4. MaldomDetector Implementation

4.1 *Raw dataset collection*

A labelled dataset is required to train the classifier of MaldomDetector using a supervised learning method. It is important that this dataset is made up of samples from many types of DGA families, so that the classifier is trained on different types of malicious domains. The quality of the underpinning dataset is crucial to any machine learning task. A high-quality pre-labelled ground truth dataset (i.e. verified malicious and benign data) was chosen to train and evaluate MaldomDetector [14] [16] [15].

The malicious domains were collected from a known ground truth dataset, DGArchive, which had been used by previous researches [14] [15]. Real DGA-based domains that appeared on the internet have been collected in the DGArchive project [22] [23]. In this project, different DGA-based malware families, such as Locky and Cryptolocker, were analysed comprehensively, and all the possible domain names generated dynamically by some types of malware have been resolved or enumerated to cover the majority of real and active DGAs. In addition, a set of DGA-based domains obtained from the Bambenek consulting feeds [24], which is also a known ground truth dataset of DGA-based domain names collected by reverse engineering specific malware families observed in real traffic. Bambenek malicious domains have also been used in past work e.g., [15] and [25]. Bambenek feeds were used to create the testing dataset to check the MaldomDetector's performance on unseen examples. All of the collected domain names from DGArchive project and Bambenek feeds were labelled as malicious. The summary of the malware families employing DGA collected from DGArchive, in addition to the DGAs taken from Bambenek dataset are indicated in Table 1.

The ground truth benign domains were collected from the Alexa top domains site [26], which lists the domain names of the most visited websites on the internet. Alexa top ranked lists were often used in the preceding research works [14] [16] [18] [19] [15] [25] because they represent trustworthy sets of normal domain names. It ranks the websites based on their popularity using different criteria, such as page views and unique visitors, and provides various lists of top sites, e.g. top 500 and top 1000. We selected 85,000 domain names from the top 1 million sites to build the required benign dataset. Since the domains of Alexa are ranked based on their popularity, the first top domains were selected instead of random selection to form the ground truth benign dataset, because they are more representative of how a benign domain looks [14] [19] [15].

Table 1 Malware families employing DGA from the DGArchive and Bambenek feeds

No.	Family	Dataset source	Example	No. of samples
1	Bimetal	DGArchive	cdcd1e19e9205f369885f972a02dfd60.info	10000
2	Banjara		ckkwestnessbiophysicalohax.com	10000
3	Beep		mghhwonojhkqp4l.com	7458
4	Black hole		okjqzgzouwqonvup.ru	732
5	China		3seykl6jum4tnsd4.biz	10000
6	Cryptolocker		cpymcuvemmmevkb.com	10000
7	Dnschanger		ifhfmmatpw.com	10000
8	Dyre		ee8758b19271bfbbd4b9aff2504028c88e.cc	10000
9	Emotet		uwqjsnkcserukhse.eu	10000
10	Gameover_p2p		rcxtwbycahrozhyxwsscwjzmr.com	10000
11	Gozi		nameswiththisuseexercise.com	10000
12	Locky		xgrdauffnegeagjlh.org	10000
13	Murofet		glxmnytxfwwitw.net	10000
14	Murofetweekly		a37b38ozn60bwnqh44c59ovj26bvfwduluhrg23.info	10000
15	Necurs		cldwkehdghhtvdvlwxps.de	10000
16	Padcrypt		dmmdldalffbbodff.info	10000
17	Ramnit		hjbhskhwhncpebritli.com	10000
18	Rovnix		dztew1odwfu8qp4mv4.net	10000
19	Sphinx		oswiudtcxpxhmmwm.com	10000
20	Tinba		ctkllmvvvnbb.biz	10000

21	Geodo	Bambenek feed	vwhsgtgvtotaqbhk.eu	577
22	P2P Gameover Zeus		aydmwoxxpnrwtgfirofarwwo.org	2000
23	Post Tover Goz		1slk8fa1u26axm1qksqvd21b9mp.biz	17423

4.2 Domain Name Analysis

The domain names in the raw datasets of Table 1 were analysed to extract a set of attributes that can be used to detect DGA-based domain names. The top-level domains (TLDs) were excluded from this analysis because the DGA-based domains use the same TLDs that are used in benign domains. It is noticeable that most of the DGA-based domain names contain meaningless strings where it is difficult to pronounce or read them. The reason for this difficulty belongs to the existence of several sequential consonant or vowel letters in most malicious domain names. Table 2 displays some examples of the malicious domain names. Fig. 2 compares the frequency for the number of sequential consonant letters between the benign and DGA-based domain names. According to some sources like [27], The letter “y” is a special letter that can represent both kinds of speech sounds, i.e., vowel and consonant, depending on its position and the letters surrounding it. Since some extracted features in this research are related to the pronunciation such as the maximum number of sequential consonant and vowel letters, therefore the status of the letter “y” can affect the values of these features and hence affect the detection accuracy. Therefore, we have considered the two cases of "y" during the implementation of the experiments using RMA and we found that considering "y" as a vowel gives better accuracy.

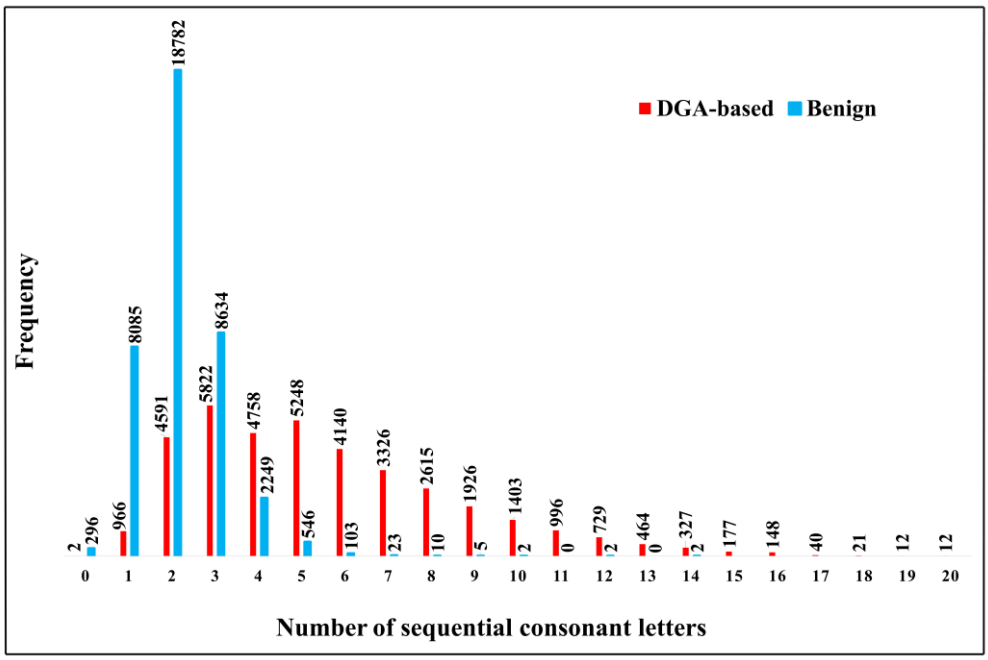


Fig. 2 The frequency of the sequential consonant letters (Benign vs DGA-based)

Table 2 Samples of DGA-based domain names

No.	Domain name	Max sequential consonant letters	Max sequential vowel letters	entropy
1	c8107dec824b212ee75fa4e5cbfcbcbb.info	8	2	3.5008
2	dnxmllaabettingk.com	6	2	3.5
3	pgghtdppldxdrr8aeo1plfyfq.com	14	3	3.8137
4	kgsyouuibangvonn.com	3	5	3.3278
5	jwkynwfxjqdqqmji.ru	11	1	3.2806

6	mwzcpkmwid.org	8	1	2.9219
7	jchhbyjryuiiuy.com	5	6	3.3927
8	vwdqvtleoyeoag.com	7	6	3.3788
9	rsxgrtkswwal.su	10	1	3.085
10	saiieiwlkcinryh.pw	4	5	3.3736
11	oxuqgbtwcnkdcwr.info	12	1	3.6402
12	dkoaeyauxjqdht.work	6	6	3.5216
13	1slk8fa1u26axm1qksqvd21b9mp	10	3	4.0141
14	a37b38ozn60bwnqh44c59ovj26bvfwduluhrg23	8	2	4.58
15	ee8758b19271bfbbd4b9aff2504028c88e	7	2	3.6321

Shannon's entropy can serve as a good metric to measure the randomness in characters distribution within a given domain name [8]. It can be calculated using equation (1).

$$H = -\sum_{i=1}^{n} p_i \log p_i \quad \ldots\ldots\ldots (1)$$

Where p_i is the probability distribution $P_n = (p_1, \ldots, p_n)$ with $p_i \geq 0$ for $i = 1, \ldots, n$ and $\sum_{i=1}^{n} p_i = 1$

Entropy exceeding a threshold value can be a useful indicator to identify DGA-based domain names. The normal distribution curves of entropy values for the DGA-based and benign domains are shown in Fig. 3. It is evident from Fig. 3 that there is a clear differentiation in the probability distribution of the entropy values between the curves, which may suggest entropy is a useful feature to classify the domains. Therefore, the entropy value for the domain name was selected as a feature to identify the DGA-based domain names. The entropy values of some domain names are indicated in Table 2.

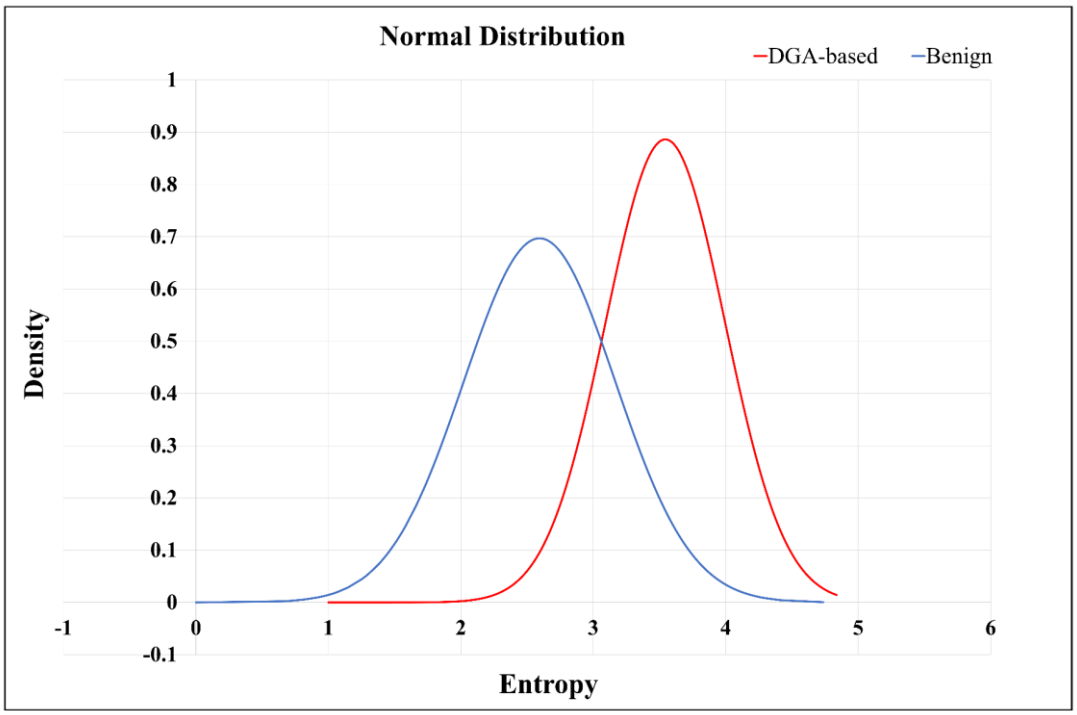


Fig. 3 The normal distribution curve for the domains: Malicious & benign

4.3 Randomness Measuring Algorithm (RMA)

RMA has been constructed to initially identify harmful domain names by measuring the randomness in the domain name's characters. It is an enhanced version of our earlier work in [28].
RMA is a deterministic algorithm that accepts a subset of the basic features as an input, i.e., the entropy, the maximum sequential consonants, the maximum sequential vowels, and the domain name length. It then processes them according to the threshold values depicted in Fig. 4. These threshold values were determined based on the domain name analysis, conducted in section 4.2.

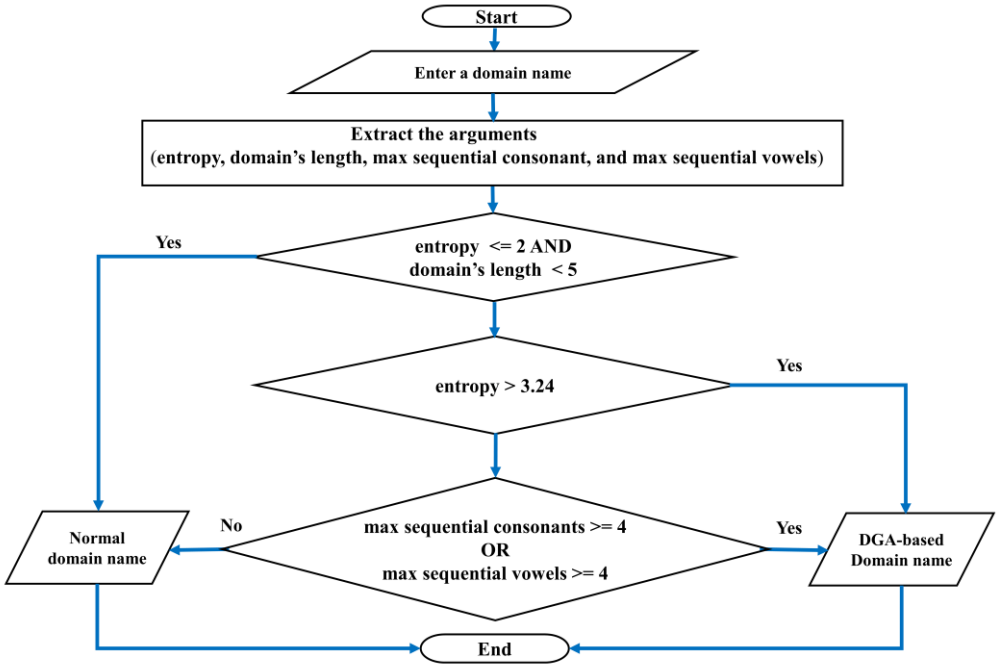


Fig. 4 The Randomness Measuring Algorithm (RMA)

After inspecting the probability distribution of the entropy values in Fig. 3, we have recognised three base points in the distribution curves that can be used in RMA as thresholds as shown in Fig. 5. The following were found:

- The ratio of the benign domains that have entropy (H) <= 2 is 18.99%, while the ratio of the DGA-based domains is 0.093%.
- The ratio of the DGA-based and benign domains that have H > 3.24 is 77.83% and 10.44% respectively.
- The values 2 < H <= 3.24 occur in the malicious and benign domains together in different proportions, making the recognition of these domains a non-trivial task, which requires additional complex feature(s). Since the goal of this research is to build a detection system using uncomplicated features, two simple features, i.e., maximum sequential consonants & maximum sequential vowels, were added to the rules of RMA, to help increase discrimination between sample distributions and reduce the false rate. The added rule states that most of the DGA-based domains have four or more sequential consonants or vowels, while many of the benign domains have fewer than four sequential consonants or vowels.

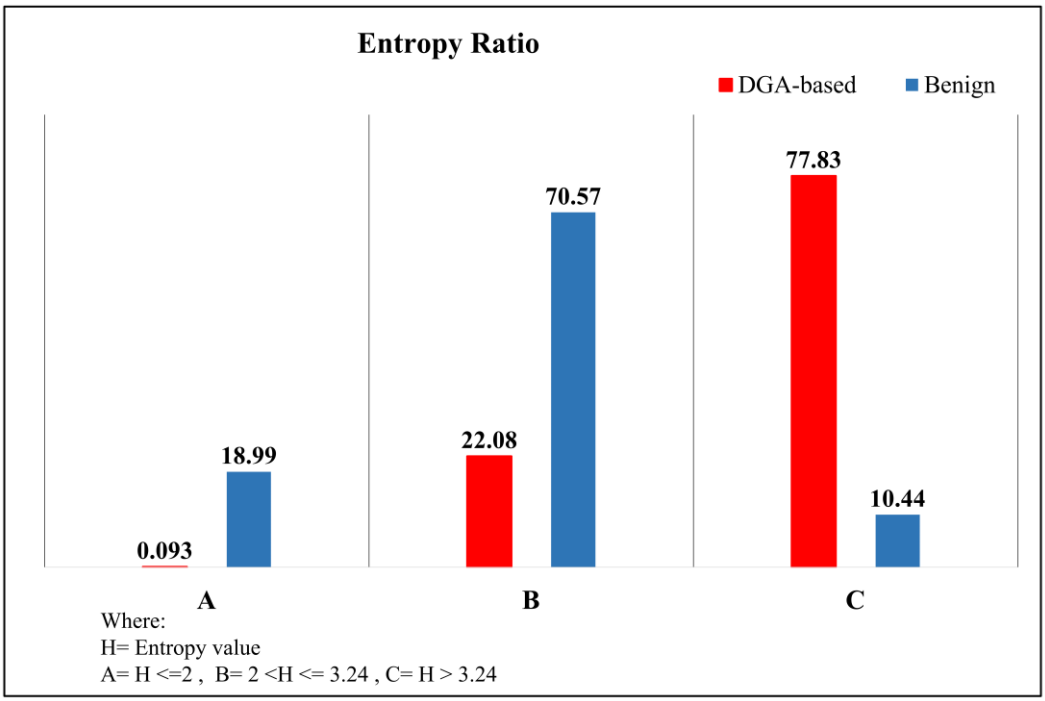


Fig. 5 Ratio of the entropy values in DGA-based and benign domains

RMA was implemented in Python and evaluated using various DGA families. RMA does not contain any parameters related to dictionary words or frequency distribution of letters; therefore, it is a language-independent algorithm. RMA has been applied to 20 types of malware families of DGArchive dataset and the results are indicated in Table 3. RMA also was applied to 85,000 of Alexa domain names where the results show detection accuracy and false positive rate are 83.14% and 0.1686 respectively.

Table 3 RMA evaluation results of DGA families and Alexa

No.	Type		Size (sample)	Classified samples		Accuracy %	False rate
				True	False		
1	DGA family	Bamital	10000	9938	62	99.38	0.0062
2		Banjori	10000	9057	943	90.57	0.0943
3		Bedep	7458	7246	212	97.1574	0.0284
4		Blackhole	732	727	5	99.32	0.00683
5		Chinad	10000	9881	119	98.81	0.0119
6		Cryptolocker	10000	9423	577	94.23	0.0577
7		Dnschanger	10000	8121	1879	81.21	0.1879
8		Dyre	10000	9988	12	99.88	0.0012
9		Emotet	10000	9900	100	99.0	0.01
10		Gameover p2p	10000	10000	0	100	0
11		Gozi	10000	7832	2168	78.32	0.2168
12		Locky	10000	8338	1662	83.38	0.1662
13		Murofet	10000	9799	201	97.99	0.0201
14		Murofetweekly	10000	10000	0	100	0
15		Necurs	10000	8957	1043	89.57	0.1043
16		Padcrypt	10000	9138	862	91.38	0.0862
17		Ramnit	10000	8970	1030	89.7	0.103
18		Rovnix	10000	9963	37	99.63	0.0037
19		Sphinx	10000	9853	147	98.53	0.0147
20		Tinba	10000	9298	702	92.98	0.0702
--		(Average)	--	--	--	94.05	0.0595
21	Benign	Alexa	85000			83.14	0.1686

Where:

$$\text{Accuracy} = \frac{number\ of\ correctly\ classified\ samples}{number\ of\ total\ samples} \quad \text{..... (2)}$$

$$\text{False rate (negative or positive)} = \frac{number\ of\ incorrectly\ classified\ samples}{number\ of\ total\ samples} \quad \text{..... (3)}$$

It is noticeable from the results above that RMA has high accuracy in detecting most DGA families and reasonable accuracy on a few types such as gozi and dnschanger. The detection process becomes difficult when a malware DGA generates domain names containing meaningful words or low randomness in their characters. However, the majority of the DGA families generate random domains because they are based on a random algorithm to generate many domains. In addition, the botmaster must register a few domains to enable the malware to make a connection. Therefore, the malware developer avoids using wordlists in algorithmically generated domains in order to avoid conflict with legitimate domains during the registration process. It also is noticeable that although some DGA families in Table 1 have domain names that contain alphanumeric characters, such as Bamital, Dyre, and Murofetweekly, they still have a number of sequential consonant and vowel letters within their characteristics and most of them have an entropy value (H) >3.24 as shown in some examples of Table 2. These families have been detected by RMA with high accuracy as indicated in Table 3.
In order to increase the detection accuracy and build a system that can address the low randomness in the domain name strings, a machine learning-based system has been constructed. MaldomDetector employs the output of RMA along with other engineering features to detect malicious domains effectively. The next section illustrates the process of building the system.

4.4 Building and evaluating the domain name classifier

4.4.1 Feature extraction and selection

Two types of features have been extracted based on the domain name analysis in section 4.2. Firstly, basic features and secondly derived features. The basic features include the entropy, max sequential consonants, max sequential vowels, the total number of consonants, the total number of vowels, and the length of a given domain name. Whereas the derived features have been calculated from the basic features, based on the domain knowledge as indicated in Table 4.

Table 4 The basic and derived features

Feature type	Feature	Feature name	Description
Basic	F1	entropy	The entropy of a given domain name string.
	F2	max-sequential-consonants	Maximum sequential consonant letters found within a given domain.
	F3	max-sequential-vowels	Maximum sequential vowel letters found within a given domain.
	F4	length-domain	Length of the domain name string.
	F5	consonants	The total number of consonant letters of a given domain.
	F6	vowels	The total number of vowel letters of a given domain.
Derived	F7	ratio-entropy-to-length-domain	The ratio of the entropy to the length of a given domain.
	F8	ratio-consonants-to-vowels	The ratio of the total number of consonant letters to the total number of vowel letters of a given domain.
	F9	ratio-consonants-to-length-domain	The ratio of the total number of consonant letters to the length of a given domain.
	F10	ratio-vowels- to-length-domain	The ratio of the total number of vowel letters to the length of a given domain.
	F11	ratio-max-sequential-consonants-to-length-domain	The ratio of the maximum sequential consonant letters to the length of a given domain.
	F12	ratio-max-sequential-vowels-to-length-domain	The ratio of the maximum sequential vowel letters to the length of a given domain.

	F13	ratio-max-sequential-consonants-to consonants	The ratio of the maximum sequential consonant letters to the total number of consonants of a given domain name.
	F14	ratio-max-sequential-vowels-to-vowels	The ratio of the maximum sequential vowel letters to the total number of vowels of a given domain.
	F15	ratio-max-sequential- consonants-to-max-sequential-vowels	The ratio of the maximum sequential consonant letters to the maximum sequential vowel letters of a given domain.
	F16	Randomness	The output of RMA algorithm.

Feature importance was calculated for all features in Table 4. A standard technique for calculating feature importance is to use the correlation, which is formally referred to as Pearson's correlation coefficient (PCC) [29]. The formula of PCC for two variables is indicated in the following equation.

$$PCC = \frac{cov\ (x,y)}{(\sigma x\ \sigma y)} \quad \ldots\ldots\ldots (4)$$

Where:

cov is the covariance of the two variables x, y.
σ is the standard deviation of the variable, which represents the square root of its covariance.
The covariance (*cov*) can be used to measure the linear relationship between two variables, i.e., it tells us how much the two variables vary together. In this work, the variable x can be any features of Table 4, whereas the variable y means the class.
We calculated PCC between each extracted feature in Table 4 and the class (response) using the SciPy library of Python. The rank and score of the importance of each feature are shown in Fig. 6. The feature importance scores can be used to reduce the number of extracted features by selecting those that have an importance score greater than a specific value (threshold) to be selected features. Since there is no certain rule to assign the threshold [30], we determined the threshold with 0.2 in this case. Therefore, the features that have an importance score greater than 0.2, i.e., F16, F7, F4, F5, F1, F2, F10, F12, F15, F8, F6, and F3 have been selected. While the features F9, F11, F14, and F13 have been eliminated because their score is less than 0.2 as indicated in Fig. 6. Since the features F16, F7, F4, F5, F1, and F2 have an importance score greater than 0.5, they have been considered the most important features for our problem. Table 5 shows the selected features that will be used to build the classifier.

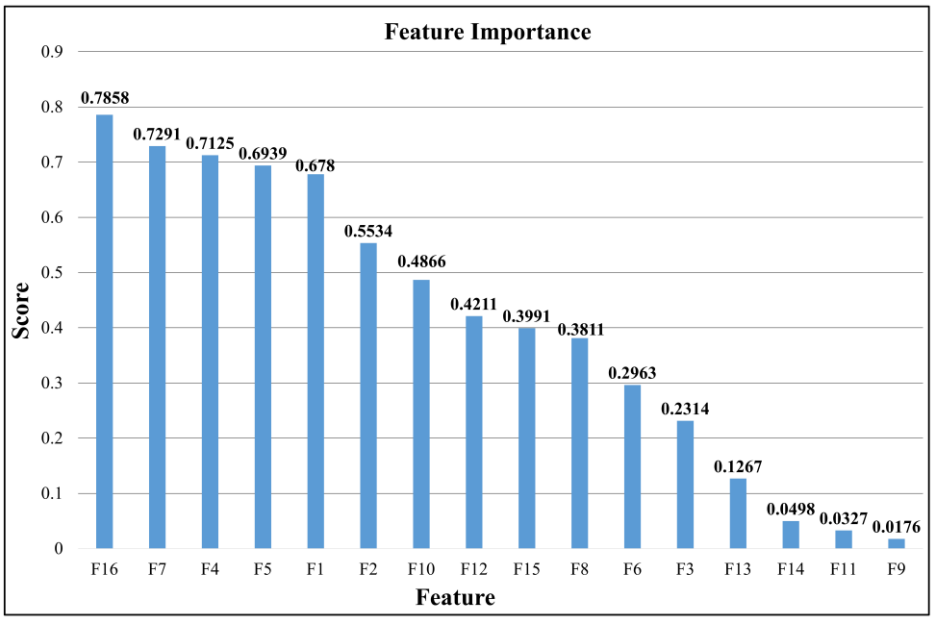


Fig. 6 The plot of the feature importance

Table 5 The selected features

No.	Feature	Feature name
1	F1	entropy
2	F2	max-sequential-consonants
3	F3	max-sequential-vowels
4	F4	length-domain
5	F5	consonants
6	F6	vowels
7	F7	ratio-entropy-to-length-domain
8	F8	ratio-consonants-to-vowels
9	F10	ratio-vowels- to-length-domain
10	F12	ratio-max-sequential-vowels-to-length-domain
11	F15	ratio-max-sequential- consonants-to-max-sequential-vowels
12	F16	Randomness

4.4.2 Building a labelled dataset

We have written several python modules to build the required dataset for training and evaluating the classifier. The main module extracted the values of the features given in Table 4 out of many malicious and benign domain names selected from the raw datasets in Table 1 and saved them in CSV files. Thereafter, the dataset was cleaned through handling the incorrect data and removing duplicates. A group of unduplicated samples was selected from each DGA family of DGArchive in Table 1 and labelled as malicious domains to create the malicious dataset. While a set of domains was selected from the Alexa data and labelled as benign domains to create the benign dataset. These malicious and benign datasets have the same number of samples and were combined to form the final dataset required to build the classifier. Table 6 summarizes the details of the dataset.

Table 6 Dataset summery

Raw dataset name	Type	No. of samples	Dataset size
DGArchive (20 DGAs of malware families)	Malicious	85000	170000
Alexa (top sites)	Benign	85000	

4.1.3 Model training

The classification learner app of MATLAB R2019a [31] was used to train the domain name classifier of Fig.1 using the dataset of Table 6. The k-fold cross-validation test option was chosen to train and evaluate the selected machine learning algorithms in order to protect against overfitting and provide an accurate model performance estimation [32] [33]. The default value of k in the classification learner is 5, however, it was set to 10 as it provided optimal accuracies. At first, we explored all the learning classification algorithms that exist in the classification learner, such as decision tree and support vector machine (SVM), to train the model using all the features in Table 5. The hyperparameters of the selected algorithms were tuned manually to improve their performance, however, we found that the algorithms produce the best performance when the default hyperparameters setting were adopted. After training a set of models using the default setting of the hyperparameters, five best models, i.e., Decision tree (Fine Tree), Ensemble (Boosted Tree), Naïve Bayes (Gaussian), and KNN (Coarse), were selected based on some evaluation metrics. Fig. 7 shows the training of the models using the classification learner app of MATLAB.

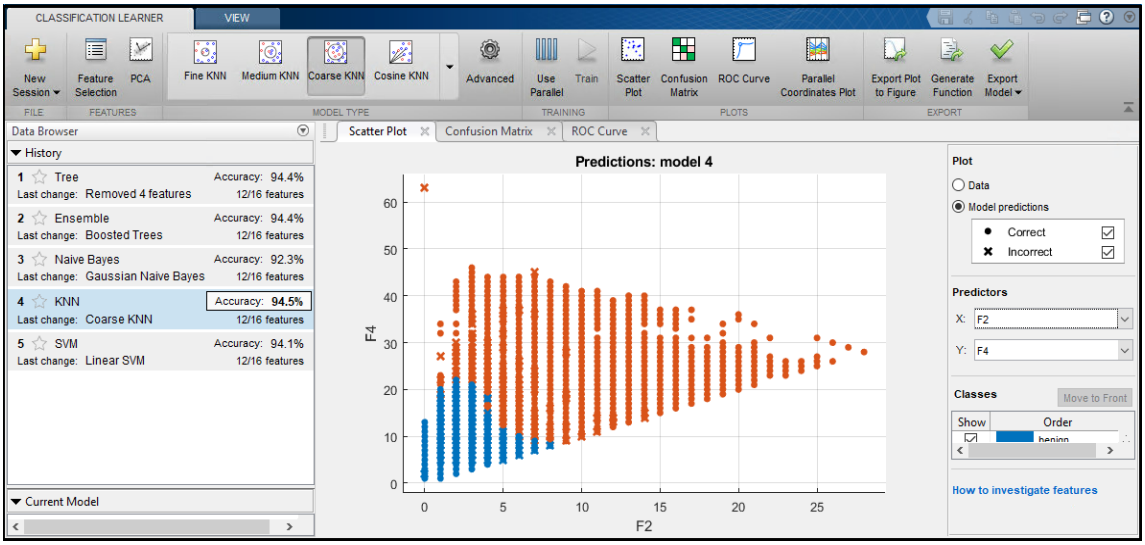


Fig. 7 Models training using MATLAB

4.4.4 Evaluating the classifier

Several binary classification metrics [34], such as accuracy, False Positive Rate (FPR), precision, recall and F1 score, were used to evaluate the performance of the trained machine learning models. These metrics can be derived from the confusion matrix and defined as shown in the equations below. The evaluation results of the best models that were selected after performing the training and validation task are indicated in Table 7, and the models' performance is depicted in Fig. 8.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$\text{FPR} = \frac{FP}{FP+TN} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

$$\text{Recall (sensitivity)} = \frac{TP}{TP+FN} \quad \ldots\ldots\ldots\ldots\ldots\ldots (8)$$

$$\text{F1 score} = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad \ldots\ldots\ldots\ldots\ldots (9)$$

Where: TP is True Positive, FP is False Positive, TN is True Negative, and FN is False Negative.

Table 7 The evaluation results of the models

Algorithm name	Hyperparameter	Accuracy (10-fold cross validation) %	FPR	Precision	Recall	F1 score
Decision tree (Fine Tree)	- Max no. of splits: 100 - Split criterion: Gini's diversity index - Surrogate decision splits: off	94.39	0.0533	0.9464	0.9410	0.9437
Ensemble (Boosted Tree)	- Ensemble method: Ada boost - max no. of splits: 20 - No. of learners: 30 - Learning rate: 0.1	94.42	0.0545	0.9454	0.9429	0.9441
Naïve Bayes (Gaussian)	Distribution names: Gaussian	92.33	0.0816	0.9192	0.9282	0.9236
SVM (Linear)	- Kernel function: linear -Box constraint level:1 - Kernel scale: auto	94.14	0.0536	0.9459	0.9363	0.9411
KNN (Coarse)	- No. of neighbours: 100 - Distance metric: Euclidean - Distance weight: equal	94.52	0.0402	0.9586	0.9305	0.9443

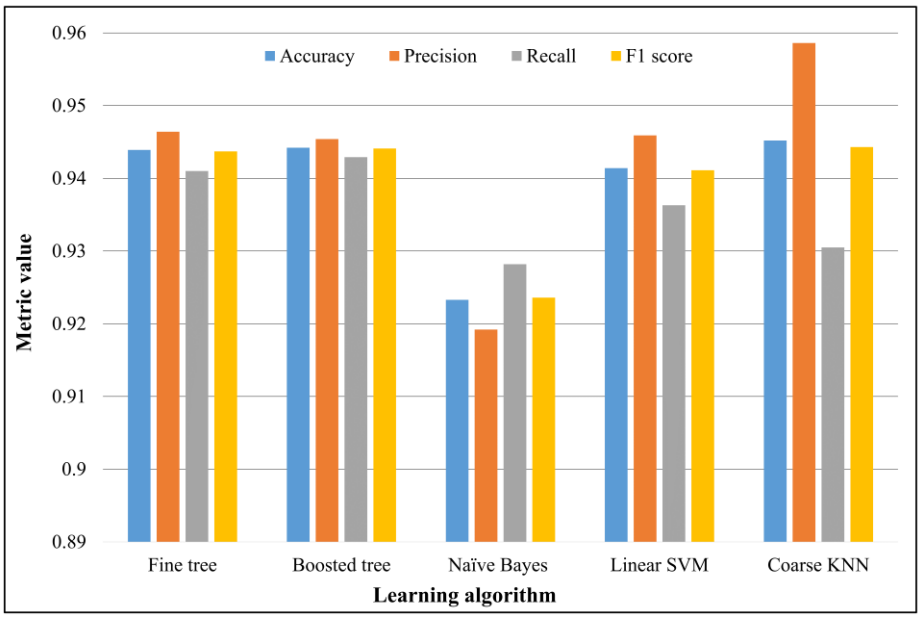
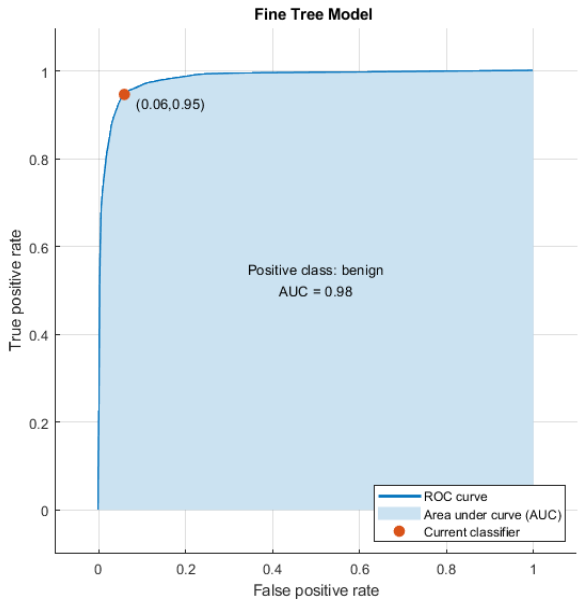
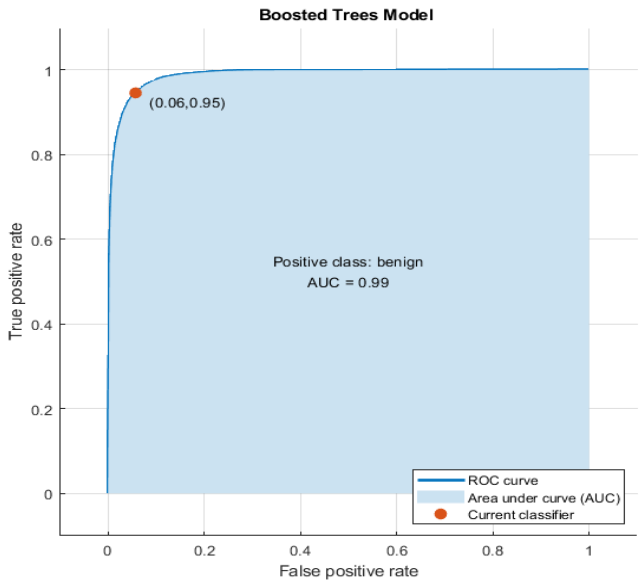


Fig. 8 Performance metric of the models

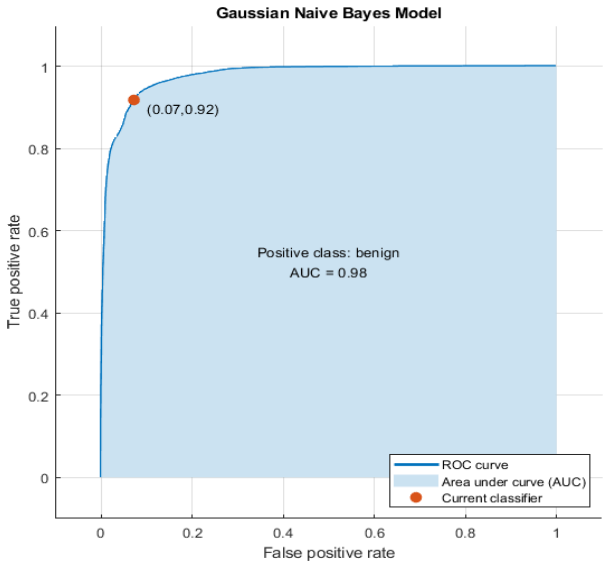
The Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) for the selected models are shown in Fig. 9. These curves can be used to assess the performance of the model over the entire operating range. It is noticeable from Fig. 9 that the models have approximately similar performance.



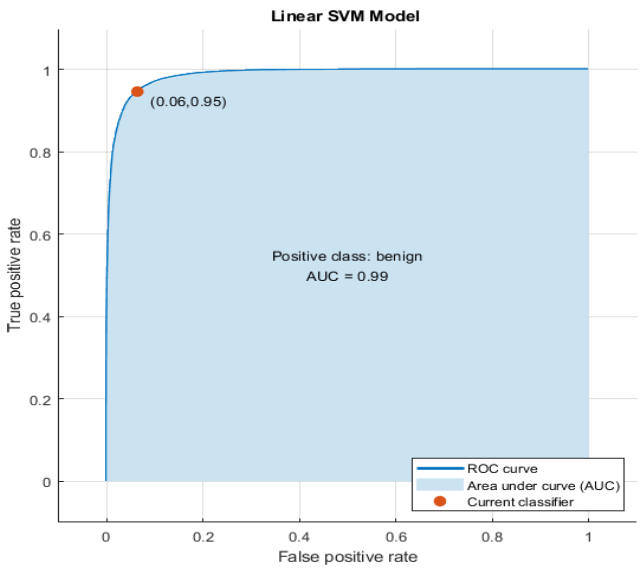
(a)



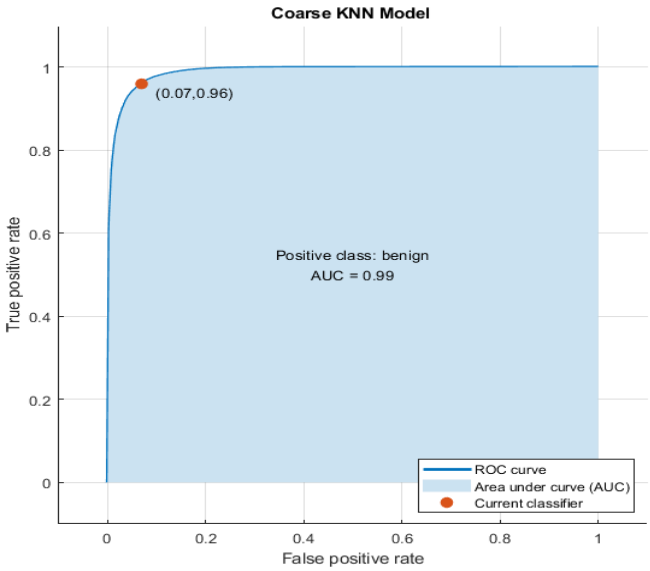
(b)



(c)



(d)



(e)

Fig. 9 ROC curve and AUC of the selected models: (a) Fine Tree (b) Boosted Tree (c) Gaussian Naïve Bayes (d) Linear SVM (e) Coarse KNN

Although applying cross validation procedure is considered enough to evaluate the performance of the models [35] [34], we made an additional assessment for the models' performance using a second dataset, i.e., Bambenek feeds, that was not used during the models building process. Some DGA families were selected from Bambenek dataset to make the evaluation. Whereas the benign domains were collected from the dataset which was used in [16] and they differ from the benign domains of Table 6. Table 8 shows the summary of the testing dataset, while Table 9 illustrates the extra evaluation results of the models. The results demonstrate that MaldomDetecor is a reliable and efficient system for detecting DGA-based domains.

Table 8 Testing dataset

Raw dataset name	Type	No. of samples	Size
• Geodo DGA • P2P Gameover Zeus DGA • Post Tover Goz DGA	Malicious	8500	17000
clean-alexa-32k	Benign	8500	

Table 9 Testing results of the selected models

Algorithm name	Accuracy %	FPR	Precision	Recall	F1 score
Decision tree (Fine Tree)	97.21	0.0542	0.9485	0.9985	0.9728
Ensemble (Boosted Tree)	97.19	0.0556	0.9473	0.9994	0.9726
Naïve Bayes	96.26	0.0738	0.9312	0.9989	0.9639
Linear SVM	97.38	0.0508	0.9516	0.9984	0.9744
Coarse KNN	97.82	0.0406	0.9609	0.9969	0.9786

5- Discussion

Since the datasets used in previous research works are not identical and there is no standard DGA to generate domain names for comparison, therefore, it is difficult to compare MaldomDetector with the systems presented in the related work section. On the other hand, the variety of DGA implementations create wide fluctuation in the detection results when applying the detection methods on these DGAs. In this section, we will only summarise some characteristics of 11 detection methods discussed in section 2 as illustrated in Table 10. As shown in Table 10, MaldomDetector has several advantages while it keeps high accuracy.

Table 10 The properties of the DGA-based domains detection methods

No.	Method name	Source	Accuracy %	Language Independency	External Source Independency	Does not need DNS response
1	Knowledge based random forest algorithm	[20]	N/Av	×	✓	✓
2	Hidden Markov Model (HMM)	[17]	91.52	✓	✓	×
3	Handcrafted features based J48	[21]	92.3	×	✓	✓
4	Handcrafted features based	[18]	N/Av	×	✓	✓
5	Implicit features based (deep neural network)	[18]	N/Av	✓	✓	✓
6	Extreme machine learning	[19]	96.29	✓	×	×
7	Machine learning based on masked n-grams	[16]	98.91	×	✓	✓
8	Deep neural network	[15]	> 98.0	×	✓	✓
9	Proposed n-CBDC (n = 2, bigram)	[14]	94.15	×	✓	✓
10	Proposed n-CBDC (n = 3, trigram)	[14]	98.29	×	✓	✓
11	MaldomDetector	this paper	97.82	✓	✓	✓

The approaches [14] [15] [16] [20] used a probabilistic language model, i.e., n-gram, which assigns probabilities to every n-sequence of characters. It estimates these probabilities by calculating the relative frequency for each n-sequence of characters within a given dataset. However, this makes the model very dependent on the training dataset and inefficient in dealing with new types of DGA-based domain names [36] [37]. [21] depended on the frequency distribution of alphanumeric letters of the domain names while [18] (Handcrafted features-based) used a dictionary matching score to measure the degree that a word in a domain name can be explained by a dictionary. MaldomDetector depends on a set of pronunciation-based features that do not depend on the training dataset and it does not adopt any probabilistic language model, i.e., language-independent system.

The method employed in [19] also used two features that require access to external site, i.e., WHOIS lookup service, to get data before detecting the malicious domains. However, getting this information adds a time delay and requires the system to be always online to function properly.

The methods in [17] and [19] require information from the DNS response packets such as TTL (time to live) and a number of domain name servers before making a decision. Although this information can be useful to reduce the false positive rates, it adds a time delay that can be exploited by the malware to contact its C&C server or exfiltrate information before detection.

The objective of this research is to build a detection system that requires little information to check the status of the requested domain names, while trying to detect malicious communications early and thus reducing risk to the network. Therefore, MaldomDetector does not use any data from an external site or from the DNS response message to classify the domains, even if they are possibly useful in the detection process. MaldomDetector has been built to depend solely on a deterministic algorithm and computationally inexpensive features extracted out of the DNS request message. This enables the system to check the domain names before sending them to the DNS server to resolve them, as a first layer of detection.

6- Conclusion

This paper presents an effective detection system, MaldomDetector, to detect algorithmically generated malicious domain names. MaldomDetector employs an algorithm, i.e., RMA, to measure the randomness in the domain name characters. MaldomDetector feeds RMA's outputs along with a set of basic and derived features extracted out of the initial DNS request to a machine learning classifier for processing and classification. Several classification algorithms have been explored to build the classifier.

Building upon the state-of-the-art on malicious domain name detection, MaldomDetector does not employ any probabilistic language models. Rather, it employs a character-based approach to detect DGA-based domain names. It performs measurements solely on the DNS request packet and does not need to wait for the DNS response to extract extra features or require information from any external sources. The evaluation results show that MaldomDetector can detect effectively different types of DGA-based domains generated by several types of malware and the detection accuracy is ~ 98%. MaldomDetector can be used to raise early alarms about potential malicious DNS communications while maintaining high accuracy.

References

[1] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A Survey on Malicious Domains Detection through DNS Data Analysis," vol. 51, no. 4, 2018.

[2] J. Gardiner, M. Cova, and S. Nagaraja, "Command & Control: Understanding, Denying and Detecting," *arXiv.org*, vol. cs.CR, no. February, p. 38, 2014.

[3] S. Yadav, S. Member, A. Kumar, 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, pp. 1663–1677, 2012.

[4] T. Wang, H. Lin, W. Cheng, and C. Chen, “DBod : Clustering and detecting DGA-based botnets using DNS traffic analysis,” *Computers & Security*, vol. 64, pp. 1–15, 2017.

[5] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, “A LSTM based framework for handling multiclass imbalance in DGA botnet detection,” *Neurocomputing*, vol. 275, pp. 2401–2413, 2018.

[6] K. Alieyan, A. Almomani, A. Manasrah, and M. M. Kadhum, “A survey of botnet detection based on DNS Article in Neural Computing and Applications · December 2015,” *Neural Computing and Applications*, no. December 2016, 2015.

[7] M. Zago, M. Gil Pérez, and G. Martínez Pérez, “Scalable detection of botnets based on DGA: Efficient feature discovery process in machine learning techniques,” *Soft Computing*, 2019.

[8] D. K. Vishwakarma, “Domain Name Generation Algorithms,” Masaryk University, 2017.

[9] B. Yu, D. L. Gray, J. Pan, M. De Cock, and A. C. A. Nascimento, “Inline DGA Detection with Deep Networks,” in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 683–692.

[10] and S. S. Deepak Gujraniya, Mohammad Waseem, Balamurali AR, “Ransomware Command and Control Detection using Machine Learning,” 2019. [Online]. Available: https://blog.acalvio.com/ransomware-command-and-control-detection-using-machine-learning.

[11] Y. Zhou, Q. Li, Q. Miao, and K. Yim, “DGA-Based Botnet Detection Using DNS Traffic,” *Journal of Internet Services and Information Security (JISIS)*, vol. 3, no. 3/4, pp. 116–123, 2013.

[12] P. Arntz, “Explained: Domain Generating Algorithm,” 2016. [Online]. Available: https://blog.malwarebytes.com/security-world/2016/12/explained-domain-generating-algorithm/.

[13] Y. I. Li, K. Xiong, S. Member, T. Chin, and C. Hu, “A Machine Learning Framework for Domain Generation Algorithm-Based Malware Detection,” *IEEE Access*, vol. 7, pp. 32765–32782, 2019.

[14] C. Xu, J. Shen, and X. Du, “Detection method of domain names generated by DGAs based on semantic representation and deep neural network,” *Computers and Security*, vol. 85, pp. 77–88, 2019.

[15] B. Yu *et al.*, “Weakly supervised deep learning for the detection of domain generation algorithms,” *IEEE Access*, vol. 7, pp. 51542–51556, 2019.

[16] J. Selvi, R. J. Rodríguez, and E. Soria-olivas, “Detection of algorithmically generated malicious domain names using masked N-grams,” *Expert Systems With Applications*, vol. 124, pp. 156–163, 2019.

[17] P. Lv, L. Bai, T. Liu, Z. Ning, J. Shi, and B. Fang, “Detection of Malicious Domain Names Based on Hidden Markov Model,” in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, 2018, pp. 659–664.

[18] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, “DGA Botnet Detection Using Supervised Learning Methods,” in *International Symposium on Information and Communication Technology*, 2017, pp. 211–218.

[19] Y. Shi, G. Chen, and J. Li, “Malicious Domain Name Detection Based on Extreme Machine Learning,” *Neural Processing Letters*, vol. 48, no. 3, pp. 1347–1357, 2018.

[20] W. Song and B. Li, “A Method to Detect Machine Generated Domain Names Based on Random Forest Algorithm,” in *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, 2016, pp. 509–513.

[21] Dinh-Tu Truong and Guang Cheng, “Detecting domain-flux botnet based on DNS traffic

features in managed network," *SECURITY AND COMMUNICATION NETWORKS*, pp. 2338–2347, 2016.

[22] F. F. K. Y. Daniel Plohmann, "A Comprehensive Measurement Study of Domain Generating Malware," in *25th USENIX Security Symposium*, 2016, pp. 263–278.

[23] D. F. F. PLOHMANN, "DGArchive." [Online]. Available: https://dgarchive.caad.fkie.fraunhofer.de. [Accessed: 07-Oct-2019].

[24] "Master Feed Listing." [Online]. Available: https://osint.bambenekconsulting.com/feeds/. [Accessed: 26-Mar-2019].

[25] B. Yu and A. Nascimento, "Character Level based Detection of DGA Domain Names," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[26] "Alexa top 1 million." [Online]. Available: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip. [Accessed: 29-Mar-2019].

[27] M. Webster, "The Truth About 'Y': It's Mostly a Vowel." [Online]. Available: https://www.merriam-webster.com/words-at-play/why-y-is-sometimes-a-vowel-usage.

[28] A. O. Almashhadani, M. Kaiiali, S. Sezer, and P. O. Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System : A Case Study of Locky Ransomware," *IEEE Access*, vol. 7, pp. 47053–47067, 2019.

[29] R. T. Gareth James, Daniela Witten, Trevor Hastie, *An Introduction to Statistical Learning*. 2013.

[30] J. Brownlee, "Feature Importance and Feature Selection With XGBoost in Python," 2019. [Online]. Available: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/. [Accessed: 23-Jan-2020].

[31] MathWorks, "Classification Learner App." [Online]. Available: https://www.mathworks.com.

[32] Benjamin S. Baumer Daniel T. Kaplan Nicholas J. Horton, *Modern Data Science with R*. CRC Press, 2017.

[33] G. Drakos, "Cross-Validation." [Online]. Available: https://towardsdatascience.com/cross-validation-70289113a072.

[34] D. Sarkar, R. Bali, and T. Sharma, *Practical Machine Learning with Python*. Bangalore, Karnataka, India: Apress, 2018.

[35] MathWorks, "Assess and improve predictive performance of models." [Online]. Available: https://www.mathworks.com/discovery/cross-validation.html.

[36] S. Kapadia, "Introduction to Language Models: N-Gram," 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9.

[37] P. K. and H. S. Nadir Durrani, Helmut Schmid, Alexander Fraser, "The Operation SequenceModel—Combining N-Gram-Based and Phrase-Based Statistical Machine Translation," *Computational Linguistics*, vol. 41, no. 2, pp. 185–214, 2015.