DGA Domain Name Detection and Classification Using Deep Learning Models

Ranjana B Nadagoudar[1], M Ramakrishna[2]
Dept. of Computer Science and Engineering, Visvesvaraya Technological University, Belagavi, India[1]
Dept. of Computer Science and Engineering, Vemana Institute of Technology, Bangalore, India[2]

***Abstract*—In today's cyber environment, modern botnets and malware are increasingly employing domain generation mechanisms to circumvent conventional detection solutions reliant on blacklisting or statistical methods for malicious domains. These outdated methods prove inadequate against algorithmically generated domain names, presenting significant challenges for cyber security. Domain Generation Algorithms (DGAs) have become essential tools for many malware families, allowing them to create numerous DGA domain names to establish communication with C&C servers. Consequently, detecting such malware has become a formidable task in cyber security. Traditional approaches to domain name detection rely heavily on manual feature engineering and statistical analysis, with classifiers designed to differentiate between legitimate and DGA domain names. In this study, we propose a novel approach to classify and detect algorithmically generated domain names. The deep learning architectures, including LSTM, RNN and GRU are trained and evaluated for their effectiveness in distinguishing between legitimate and malicious domain names. The performance of each model is evaluated using standard metrics such as precision, recall, and F1-score. The findings of this research have significant implications for cyber security defense strategies. Our experimental findings illustrate that the proposed model outperforms current state-of-the-art methods in both DGA domain name classification and detection. Our proposed model achieved 99% accuracy for DGA classification. By integrating additional feature extraction and knowledge-based methods our proposed model surpasses existing models. The experimental outcomes suggest that our proposed model gated recurrent unit can achieve 99% accuracy, a 94% recall rate, and a 98% F1-score for the detection and classification of DGA-generated domain names.**

Keywords*—Botnet; cyber security; Domain Generation Algorithms (DGAs); gated recurrent unit; Domain Name System (DNS)

I. INTRODUCTION

With rapid advancement in the information technology and development of mobile Internet, there is increase in the Internet connected devices, detecting malicious domain names is of great importance for network security. The rapid evolution of the Internet has revolutionized daily life with unprecedented convenience, yet it also presents a formidable threat to network security. A constant stream of malicious attacks continues to emerge, with botnet-based attacks standing out as a major concern [1]. Botnets consist of a chain of malware-infected hosts, where a central machine commands these compromised hosts remotely via a C&C server to carry out malicious activities. Utilizing the Domain Generation Algorithm (DGA), botnets exploit algorithmic characteristics to generate pseudo-random strings and dynamically select connected hosts, significantly enhancing their stealth and resilience [2]. Consequently, detecting DGA domain names with high accuracy and minimal cost is crucial for safeguarding network security.

Conventional methods for detecting domain names primarily rely on extracting artificial features from Domain Name Server (DNS) traffic or statistical characteristics of domain name language. Machine learning is then applied to analyze these features for the classification and identification of domain names [4]. However, accurately identifying the appropriate type of DGA is a challenging task. Each DGA family typically represents a cluster of similar algorithms, and various types of DGAs exhibit distinct DNS traffic patterns and statistical characteristics of domain names. Consequently, detection strategies that hinge on artificial feature extraction are costly and lack adaptability, rendering them inadequate for handling the intricacies of DGA types [5]. Therefore, the development of a DGA detection model using deep learning has garnered research attention as an enhanced detection approach compared to traditional methods.

Developing cyber security solutions remains a formidable challenge. Traditional signature-based detection systems rely on human involvement in continuously oversee and revise signatures, making them ineffective against emerging forms of cyber threats and emerging malware. Recent advances in optimization and parallel/distributed computing technologies have enabled the efficient training of large-scale datasets. Deep learning, a subset of artificial intelligence, has significantly improved performance across various domains [6]. Architectures like LSTM, RNN and GRU have demonstrated superior performance in cyber security applications compared to classical machine learning algorithms.

Real-time approaches for detecting Domain Generation Algorithms (DGAs) aim to classify domains as either benign or generated by a DGA [7]. Retrospective methods have shown poor performance in this regard. Early detection techniques likely employed machine learning methods. Classical approaches to machine learning-based DGA detection heavily emphasize feature engineering, which results in performance of these methods are dependent on domain specific features [8]. Recently the deep learning architectures have been considered for DGA classification and detection, these methods perform better over the traditional ML algorithms, which circumvent feature engineering and shows significant performance improvement [12].

In this paper, our intent is to evaluate the effectiveness of deep learning models for algorithmically generated domain detection. We suggest a deep learning technique called gated recurrent unit for DGA classification and detection. Initially this model performs the binary classification, which gives the probability of being benign or DGA generated. Further it detects whether the domains are legitimate or DGA generated, if the domain name is generated by a DGA then it will categorize the domain into respective DGA family it belongs.

The structure of this paper is as follows. Section II examines algorithmically generated domain names and reviews related work on DGA domain detection. Section III provides an explanation of the domain generation algorithm. Section IV covers the theoretical background of deep learning methods. Section V outlines the overall procedure for DGA domain classification and detection, while Section VI describes the dataset used. Section VII evaluates the detection performance of the proposed deep learning models through experimentation. Results and discussion is given in Section VIII. Finally, Section IX presents the conclusions drawn from the study.

II. Related Work

In recent years, many malware families have shifted their approach to communicating with remote servers. To distinguish DGA-generated domains from normal ones, researchers have identified distinct features associated with DGA-generated domain names. Consequently, numerous studies focus on blocking these DGA domain names as a defensive measure [2]. Traditional malware control methods, such as blacklisting, static string-matching approaches, and hashing schemes, are insufficient for addressing DGA threats. Several researchers have worked on algorithmically generated domain name detection some of these research papers are discussed below.

Mathew [3] devised a classification system for domain generation algorithms based on DNS traffic, along with presenting various detection techniques for DGA botnets. Among these techniques, the genetic algorithm for DGA detection was proposed. However, this method is hindered by computational complexity and high implementation costs.

Daniel Plohmann [4] presented a comprehensive study of domain generation algorithms as they are used by modern botnets. This study uses the reverse engineering of the DGAs of 43 malware families and their variants [5]. The author performed an analysis on domain registrations, utilizing historic WHOIS data. They have characterized the registration behavior of bot masters and sink holes and examined the effectiveness of domain mitigations [6]. The author explained the complexity of word list-based DGA families and their detection.

Tong and Nguyen [7] employed semantic indicators like entropy, domain level, frequency of N-grams and Mahalanobis distance were utilized in domain classification [8] for detecting DGA domain names. They proposed resampling as a preventive measure at the data level, categorizing it into oversampling, under sampling, and hybrid sampling combining both approaches.

K. Alieyan [9] introduced a rule-based schema for the domain name system designed to identify inconsistencies within it. The results of the study demonstrated an accuracy of 99.35% in detecting botnets, accompanied by a low false positive rate of 0.25. It should be noted that this approach is specifically tailored for DNS-based traffic flows.

Kheir et al. [11] introduced the Mentor method, which gathers statistical features from suspicious domain names and employs supervised machine learning to distinguish between benign and suspicious domains. This method was tested against an extensive collection of public botnet blacklists. The results indicate that the Mentor system effectively detects malicious bots while maintaining a low false positive rate by filtering out benign domain names.

The author’s Woodbridge et al. [12] presented a method that makes use of LSTM to categorize the DGA-generated and legitimate domains. LSTMs have benefits over other approaches as they are not dependent on features and make use of raw domain names as their input [14]. The experimental results show that LSTM outperformed as compared with random forest with manually engineered features and logistic regression with bigram features [16]. These approaches can perform real-time detection but they are sensitive to the imbalanced dataset which makes it difficult to detect domains from minority families.

Cheng [18] conducted an analysis comparing legal domains with DGA domain names, identifying significant deviations in domain name construction rules. They utilized domain name length and character information entropy as classification features for detecting DGA domain names. Y. Li et al. [19] investigated the distribution of alphanumeric characters and bigrams across domains sharing the same set of IP addresses to analyze the statistical characteristics of domain name language. They also evaluated the efficacy of various distance metrics in this context.

Lison et al. [25] similarly adopted an approach where they substituted the LSTM layer with a GRU layer, achieving an AUC of 0.996. Meanwhile, Mac et al. [26] employed additional embedding and integrated an LSTM with an SVM, as well as a bidirectional LSTM, achieving AUCs of 0.9969 and 0.9964, respectively, on comparable datasets.

To address the constraints of machine learning techniques in the aforementioned scenarios Curtin et al. [30] designed a framework for detecting DGA domains with recurrent neural networks. The author has presented a complexity for domain name families called the smash word score; it quantifies how much DGA domain is to English words. Further, the DGA families having higher smash word scores will usually pose greater difficulty for detection [29]. The author used a recurrent neural network model with logistic regression for DGA detection which outperforms the existing approaches of DGA detection. The limitation of this study is results are not up to the mark and this is not adoptable for corporate use.

However, while these studies have demonstrated high detection rates for specific DGA families, machine learning-based detection systems often perform poorly against new DGA variants when trained on unrepresentative or imbalanced datasets. Addressing this issue, Anderson et al. [31] employed a Generative Adversarial Network (GAN) to create domain names that traditional DGA classifiers struggle to identify. The generator produced synthetic data used to train new models. Initially, an auto-encoder was pre-trained on approximately 256,000 domains and subsequently fine-tuned. The new models were then trained and evaluated using these newly generated domain names [32]. As a result, models trained on the newly generated domains exhibited an overall improvement in True Positive Rate (TPR) from 68% to 70%.

Chin et al. [33] devised a machine learning-based framework for identifying and detecting domains generated by DGAs. Further they have applied the proposed ML techniques to investigate the DGA-based modern malware. The proposed model comprises two levels containing the classification as first level operation and the clustering method as a second-level operation. These methods are to detect and identify the algorithmically generated domains. In this work ML-based methods apply DNS blacklist for detecting DGA-generated domains.

Vinayakumar et al. [34], the author developed a model that gathers traffic data of DNS at the ISP level. Further, it identifies the DGA-based domains in real-time. They also used many deep learning models such as LSTM, CNN, CNN-LSTM and RNN for modern botnet detection. These methods have performed well compared to classical ML approaches and also give better classification accuracy rate.

Vinayakumar et al. [35], the author had designed and developed scalable architecture called Apache spark. The proposed model gathers DNS logs data and performed the analysis. The deep learning techniques are being used to detect and gives alert for suspicious domains.

The literature survey shows that recent methods for DGA domain name detection and classification based on machine learning performed better results and most importantly deep learning specifically recurrent based models. In this work, we apply enhanced model of LSTM called as GRU for DGA domain name detection and classification.

III. Domain Generation Algorithm

A Domain Generation Algorithm (DGA) operates by utilizing available sources of randomness within malware to generate hundreds or even thousands of domains automatically. DGAs enable malware to constantly switch between these domains during attacks, complicating efforts to block and remove them [3]. Cybercriminals and botnet operators exploit DGAs to deliver malware, ensuring continuous communication with Command and Control (C&C) servers through dynamically generated domains [5]. The malware attempts to query each domain against its local DNS server, vital for translating domain names into IP addresses on the Internet [9]. Only domains registered by the botmaster yield valid IP addresses for C&C communication; unregistered domains return resolution errors and are disregarded [10]. One prominent example is Banjori, widely recognized for targeting online banking users to steal information [11]. According to NSFOCUS, Banjori was first identified in 2013, and as of in 2019, a total of 1,499 botnets related to this issue were detected.

2019, a total of 1,499 related botnets had been detected, with numbers continuing to rise. Other DGAs, such as Tinba and Ramnit, specialize in financial theft and worm infections [12]. Given the diversity of DGA types, precise classification of these domains is crucial.

The algorithm takes DNS queries or domain names as input. If a domain is broken down into words or letters, it can be viewed as a sequence akin to a sentence. Certain words serve as the core components, representing key features of the domain name. Domain names within the same DGA botnet family exhibit similar characteristics based on shared keyword sets [13]. Consequently, domains from different families differ significantly in their specific keywords.

Domains within the same family create a consistent context based on the characteristic keyword sets used to generate them [15]. These contextual differences are essential in the classification process. Initially, query domains are processed through a trained binary classification model to distinguish between malicious and benign domains [17]. Domains identified as malicious then undergo multiclass classification to accurately determine their DGA botnet family, as illustrated in Fig. 1. The binary classification distinguishes between benign and malicious domain names, while the multiclass classification identifies the specific botnet family associated with malicious domains.

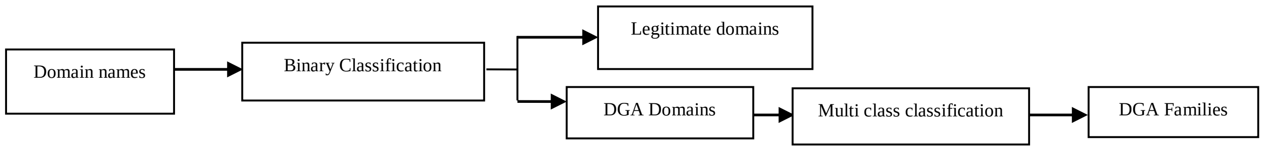

Fig. 1. Steps involved in the DGA botnet detection.

IV. Deep Learning Models

A. Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is acknowledged as an enhanced version of the standard recurrent neural network, first introduced by Cho et al. in 2014. It addresses the issue of the vanishing gradient that commonly affects traditional RNNs. GRU shares similarities with LSTM (Long Short-Term Memory) networks in design and often yields comparable performance.

GRU incorporates gating mechanisms and a hidden state to regulate information flow [20]. It tackles RNN issues by employing two gates: the update gate and the reset gate. These gates act as vector components (0, 1) capable of performing a convex combination. This mechanism determines whether to update (retain) or reset the hidden state based on incoming information [21]. Consequently, the network learns to disregard irrelevant temporal details.

GRUs are effective for enhancing the memory capabilities of recurrent neural networks and facilitating easier model training. They are widely applicable in diverse fields such as speech signal modeling, machine translation, and handwriting recognition.

The basic RNN suffers from short-term memory problems. GRU is a modified or lightweight version of LSTM, where it combines long and short-term memory into its hidden states [22]. LSTM has two attributes cell state and hidden state; here it has only a hidden state which can combine both long and short-term memory. The GRU (Gated Recurrent Unit) is characterized by its two primary gates: the update gate and the reset gate. The update gate plays a crucial role in memory retention, determining how much past information should be preserved. In contrast, the reset gate decides how much past information to discard.

The update gate is pivotal in allowing the model to selectively retain relevant earlier information across time steps. This capability is particularly advantageous as it enables the model to potentially retain all pertinent details from the past, thus mitigating the issue of vanishing gradients.

Conversely, the reset gate determines the degree to which past information should be forgotten. Similar to the Forget gate in LSTM, the reset gate identifies and disregards irrelevant data, facilitating the model's ability to progress without unnecessary baggage from previous computations. Both gates contribute significantly to the GRU's architecture and function. While their formulas are similar, their respective roles and weights within the model are distinct, as elaborated in subsequent sections.

In the GRU architecture, two key gates play crucial roles: the reset gate and the update gate. These gates are responsible for dynamically adjusting how much information each hidden unit retains or discards as it processes a sequence. In the figure illustrating the Gated Recurrent Unit, denoted as GRU, the symbols r and z represent the reset and update gates respectively, while h and h’ correspond to the activations and candidate activations. This configuration is discussed in reference [23].

During operation, when the reset gate r approaches zero, the hidden state disregards the previous state and resets based on the current input. This mechanism allows the model to efficiently discard irrelevant data, leading to a more streamlined representation. Conversely, the update gate z regulates how much information from the previous hidden state is transferred to the current hidden state. This process resembles the function of a memory cell in LSTM networks, facilitating the retention of long-term dependencies.

At any given time step, the activation of the GRU is determined through a linear interpolation between the previous activation and the candidate activation. The update gate z dictates the extent to which the unit updates its activation or content, thereby influencing the flow of information throughout the network.

V. Proposed Architecture

The proposed method includes the domain names as an input which contains series of characters, later it transforms these characters into a series of vectors. In our proposed work we have adopted Keras character level embedding. After translating the character representation into a series of vectors in the next step series of vectors are provided for deep learning layers [24]. Further, it processes the series of vectors in the sequential order and at each step it updates the hidden vector state information. Finally, the model will be able to perform the binary classification task to categorize the domains as either benign or DGA generated, where as in multi-classification task initially it detects either the domain name is benign or DGA generated. If it is DGA generated then it classifies into the corresponding malware families.

The outline of our proposed approach for detecting DGA domains using deep learning model is shown in Fig. 2. The model represents three stages of operation, first one is character encoding here it maps each character into real-valued vector and the second one is feature representation and lastly, the fully connected layer and softmax function differentiates between DGA classes, including a category for non-DGA instances. The proposed model is evaluated on both binary classification and multi-class classification that classifies whether domains are benign or DGA generated. Deep learning models are educated and evaluated on the dataset for DGA Detection.

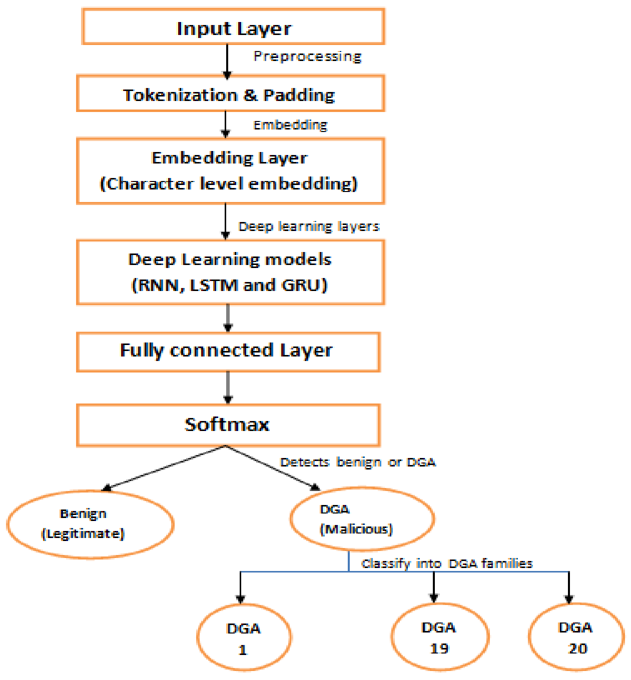

Fig. 2. Proposed deep learning architecture for DGA detection.

A. Domain Name Character Embedding

The Keras framework includes an embedding layer that translates character-level domain names into dense vectors representation shown in Fig. 3. The embedding is also learned independently during training. In this paper we have utilized keras for DGA detection [27]. In the initial phase of the operation weights are going to be assigned and later these weights will learn all the characters in the dataset. Further these embedding layer tries to maps each one of character in the dataset to a 128 length of real value represented in the vector. The domain name character level embedding makes use of recurrent neural network to determine numerical value representation by looking at their character level compositions.

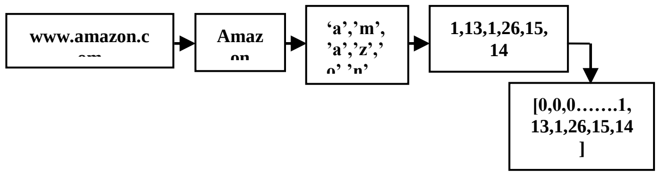

Fig. 3. Domain name character level embedding.

B. Domain Name Feature Extraction

For representing the features various deep layers have been used such as LSTM, RNN and GRU. These structures will capture the sequential information. The pattern-matching approach along with the deep learning layer looks effective and efficient compared with regular expression [28]. The regular expression outputs a binary value but the deep learning models produce a continuous value which in turn represents how much the pattern is matched.

C. Recurrent Layers

We have used many recurrent networks including LSTM, RNN and GRU. Here the number of recurrent units is set to 128 based on the knowledge acquired. These recurrent network layers primarily capture the sequential data from the output of the embedding layer. Each unit in a recurrent neural network employs an activation function with values ranging from [-1, 1]. The gate uses a logistic sigmoid function, which produces values in the range of [1, 0].

D. Deep Learning for the Binary Class Classification Task

Binary classification task involves distinguishing between two classes: benign and DGA (malicious). In our approach, benign domains are designated with a label of 0, whereas DGAs carry a label of 1. To tackle this task, we trained several deep neural network models, such as LSTM, RNN, and GRU, drawing on existing research in deep learning for character-based text classification. To optimize these neural networks for classifying domain names as either benign or malicious, we refer to the detailed descriptions of the model architectures provided in previous studies. When applying these trained neural models to a test dataset, a domain is labelled as benign if the probability is less than 0.5 and malicious if it is 0.5 or higher.

Deep Learning for the Multiclass Classification Task. The dataset used for multiclass classification contains domains from both the "benign family" and 20 distinct DGA families, totaling 21 families. For this task, we employed a model architecture similar to that used for binary classification. However, instead of two prediction classes, the models now predict among 21 classes (one class per family). Therefore, the output layer of the models from reference [18] was adjusted to use the "softmax" activation function. This ensures that the output values range between 0 and 1, representing predicted probabilities. To facilitate this, we applied one-hot encoding, resulting in 21 output values, each corresponding to a class. The class with the highest probability is selected as the final prediction made by the model.

VI. Dataset Description

The proposed domain name detection model was assessed on AmritaDGA dataset for identifying malwares/botnets from the DNS traffic [12]. AmritaDGA is a benchmark dataset publically available for research purpose. This database was used in DMD-2018 shared task and after the shared task this database has been used for benchmark purpose by various researchers for DGA detection [13]. Following, in this work, the AmritaDGA database was used for DGA domains detection. The domain name in the dataset is labeled as benign or DGA family. The dataset is further divided into training and testing respectively.

All deep learning models are trained using the training dataset. Further the dataset is comprised of training, validation and testing dataset. The training, validation and testing domain name samples are shown in below Table I.

TABLE I. Detailed Information of the Dataset

Label	Domain Type	Training	Testing	Validation
0	benign	25574	6414	8079
1	banjori	3779	1021	1165
2	corebot	3815	954	1242
3	dircrypt	3890	942	1201
4	dnschanger	3883	961	1187
5	fobber	3815	953	1203
6	murofet	3823	942	1237
7	necurs	3282	823	993
8	newgoz	3858	987	1185
9	padcrypt	3802	932	1145
10	proslikefan	3823	946	1176
11	pykspa	3834	940	1194
12	qadars	3848	972	1172
13	qakbot	3844	964	1209
14	ramdo	3907	963	1198
15	ranbyus	3868	980	1258
16	simda	3870	959	1195
17	suppobox	3850	948	1234
18	symmi	3802	952	1173
19	tempedreve	3818	932	1179
20	tinba	3845	973	1197

VII. Performance Metrics

We have adopted performance measures to compare the accuracy of various DGA classification models. Further the various metrics have been used to determine the quality of DGA classification models. AUC, recall, precision, F1 score, and ROC performance evaluation metrics are used to compare the GRU with other deep learning classification techniques.

True Positive: It represents the number of domains classified as legitimate and which is indicated with class 0.

True Negative: This represents the number of domains classified as DGA generated and which is indicated with class 1.

False Positive: It represents the number of domains wrongly classified as legitimate.

False Negative: It represents the number of domains wrongly classified as DGA generated.

$$Precision = \frac{TP}{(TP+FP)} \quad (1)$$

Recall: Measures the completeness of correctly labeled features.

$$Recall = \frac{TP}{(TP+FN)} \quad (2)$$

F1-score: Defines the harmonic mean between precision and recall measures.

$$F1 - Score = 2 \times \frac{(Recall*Precision)}{(Recall+Precision)} \quad (3)$$

Receiver operating characteristic measures the trade-off of the TPR to FPR where,

$$TPR = \frac{True\ Positive}{TruePositive+FalseNegative)} \quad (4)$$

$$FPR = \frac{FalsePositive}{(FalsePositive+TrueNegative)} \quad (5)$$

The ROC curve established with recall and false positive rate. It also shows the capability of the binary classifiers. The ROC curve also measures the competence of the classifier in differentiating the classes as either DGA or legitimate. The graph is plotted between the two metrics recall and false positive rate.

$$AUC = \int_0^1 \frac{TP}{(TP+FN)} d \frac{FP}{TN+FP} \quad (6)$$

The macro avg. and weighted avg. are used to average the results over the classes. The Macro avg. computes the elements independently and finally, it takes the average over all classes. In this paper, the weighted averaging considers as a significant performance indicator.

VIII. Results and Discussion

The deep learning techniques were executed utilizing Tensor Flow and Keras. The performance of the trained models was evaluated on a per-epoch basis using testing samples. For baseline comparison, we applied a logistic regression model to bigrams in the character-level representation of domain names, as well as other deep learning methods such as RNN, LSTM, and GRU. The experimental results demonstrate that these methods effectively classify domain names as either benign or DGA-generated in binary classification and further categorize algorithmically generated domain names into their respective malware families, based on metrics such as accuracy, precision, recall, and F1-score.

The performance of our trained model is assessed using testing samples on an epoch-by-epoch basis, as shown in Fig. 4. The baseline model showed good performance till epoch 27 and this model gives AUC of 0.988. LSTM model is evaluated using testing samples on epoch 9 which is shown in Fig. 5, whereas the RNN model performed well till epoch 11.

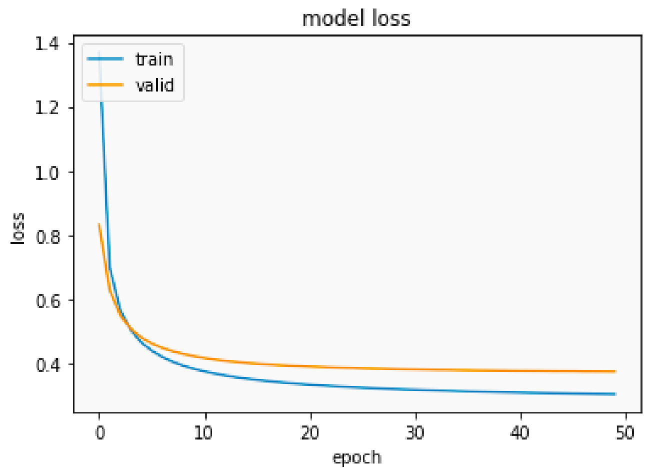

Fig. 4. Training and validation loss curve for baseline model with LR.

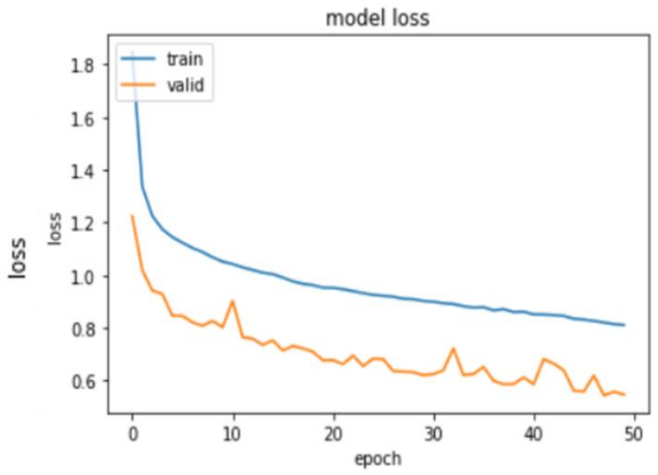

Fig. 5. The validation and training loss curve for LSTM.

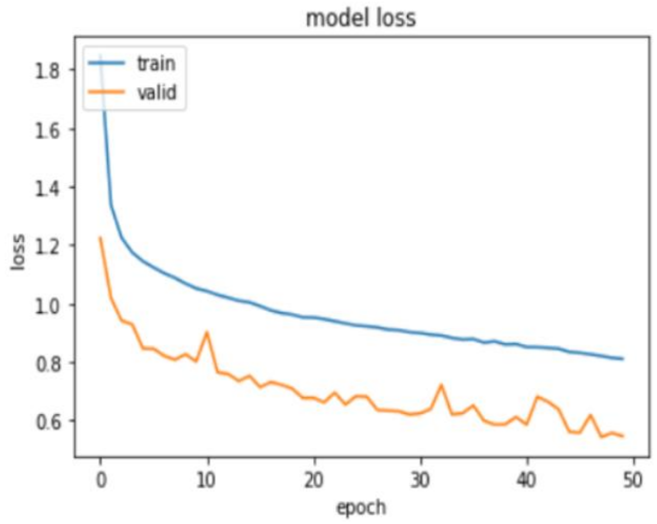

Fig. 6. Training and validation loss curve for RNN.

The proposed deep learning model gated recurrent unit has showed better performance till epochs 14 shown in Fig. 6. After that point, the performance began to decline due to over fitting. This indicates that 15 epochs are sufficient to capture the dependencies of domain names at the character level. For baseline comparison, we applied a logistic regression model to bigrams in the character-level representation of domain names.

The performance of the deep learning models is evaluated using two types of averages: weighted average and macro average. The weighted average takes into account the total number of samples by calculating the performance for each class and then averaging these performances, weighted by the sample distribution across classes. This method allows us to assess the overall performance across the entire dataset. The macro average, in contrast, involves calculating the performance for each class separately and then averaging these values, treating all classes as if they have the same number of samples. This provides the average performance per class. In cases of class imbalance in multiclass classification, the weighted average is often more representative than the macro average. Nevertheless, to obtain a comprehensive understanding of the models' performance, we compare the results of both the weighted average and macro average.

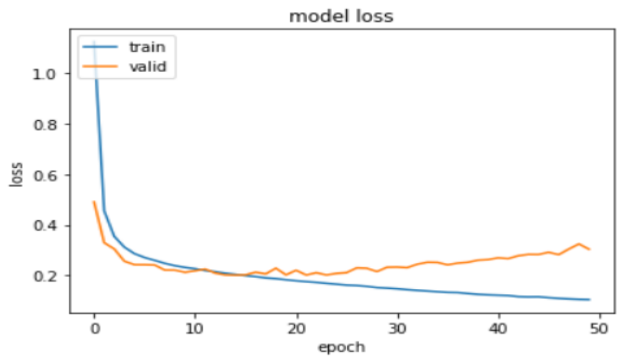

Fig. 7. The validation and training loss curve GRU.

Initially, various test experiments were run to identify various parameters for GRU model. In the proposed GRU model, the first layer is embedding and it contains the embedding length parameter. We run experiments with 32, 64, 128, and 256. The performance with 128 was good compared to others and when we increased 128 to 256, the performance remained same. Thus we decided to set the embedding length as 128. Each character of the domain will be transformed into 128 length vector. Next embedding layer follows GRU layer and again similar experiments were done and 128 units were set to GRU layer shown in Fig. 7. GRU layer follows the classification or output layer. Also, dropout was added in between the output layer and GRU layer.

The results of the models are represented in the form of a PRC curve by differentiating two parameters, such as precision and recall. The ROC curve is represented with false positive rate and recall value. Deep learning model outperforms the N-gram derived features with huge size of domain names. The PRC curves for the Bigram baseline model, LSTM, RNN and GRU are presented in Fig. 8. The performance metrics of various methods are displayed in precision recall curve which shows that the gated recurrent unit results with higher precision and recall value compared with other approaches such as RNN and LSTM, the GRU model gives in better precision and recall value shown in Fig. 9. The green line represented the Fig. 10 indicates gated recurrent unit, red line indicates LSTM and blue line indicates the baseline model. The calibration performance metrics was used to assess the performance of classifier and also to tune its parameter shown in Fig. 10. The ROC curve is represented with false positive rate and recall value shown in Fig. 11. The gated recurrent unit gives the performance with an AUC of 0.999 shown in Fig. 12.

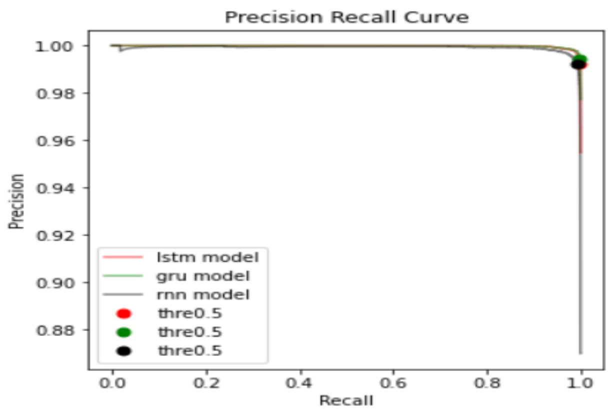

Fig. 8. Performance comparison of deep learning models.

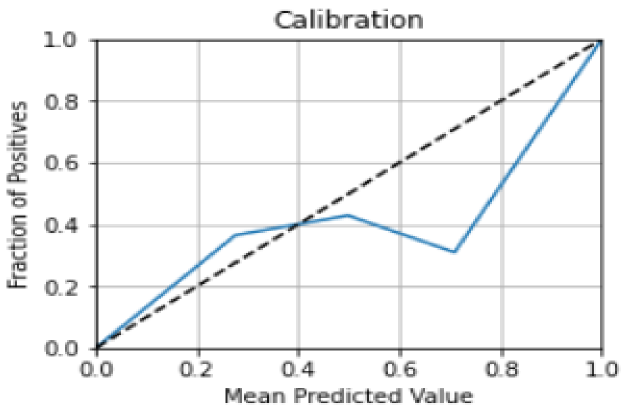

Fig. 9. Calibration.

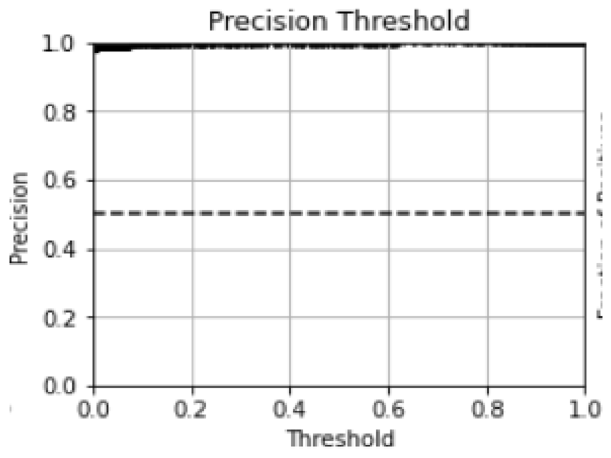

Fig. 10. Precision threshold.

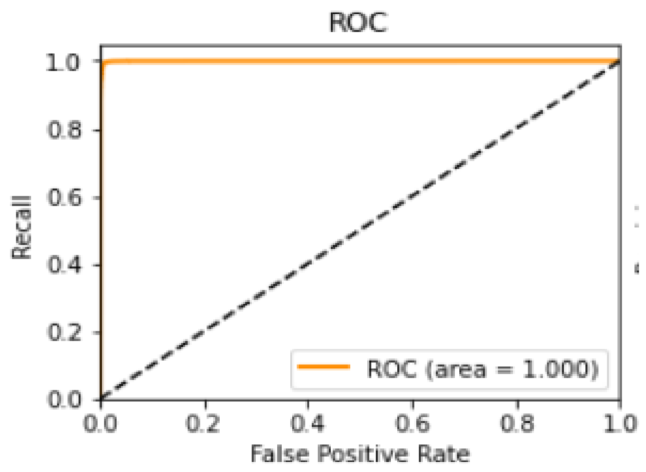

Fig. 11. Receiver operating characteristic.

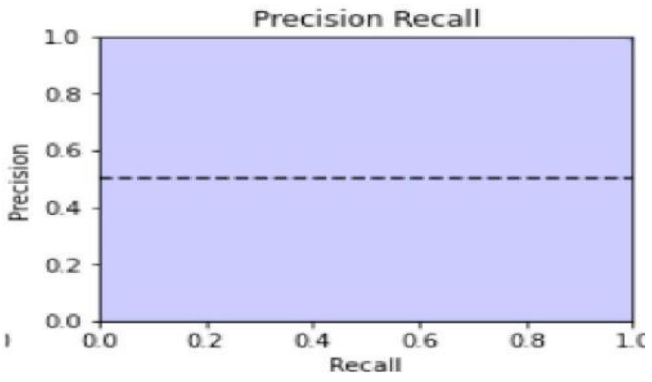

Fig. 12. Precision recall.

Table II represents the experimental results of recurrent neural network for multiclass classification of DGA domains to classify the domain name into a family of malwares in terms of accuracy, precision, recall, and f1-score.

Table III shows experimental results of deep learning approaches for multiclass classification of domain names into a family of malwares. The proposed detection model gated recurrent unit significantly outperforms in comparison with other deep learning models like LSTM and RNN in all measurements, in which our model produces accuracy of 99%, F1-score of 98%, macro avg. of 91% and weighted avg. of 92% and also provides significantly reduced false positive rate (FPR) and false negative rate (FNR). Whereas the LSTM produces accuracy of 92%, F1-score of 98%, macro avg. of 90% and weighted avg. of 91% and RNN models produces accuracy of 83%, recall of 94%, precision of 98%, macro avg. of 80% and weighted avg. of 84%. The baseline model for logistic regression produces accuracy of 87%, f1-score of 99%, precision of 98% and macro avg. of 84% and weighted avg. of 87%.

TABLE II. RESULTS OF MULTICLASS CLASSIFICATION FOR RNN

	Precision	Recall	F1-score	Support
benign	0.99	0.94	0.97	8079
banjori	0.98	0.99	0.99	1165
corebot	0.99	0.99	0.99	1242
dircrypt	0.51	0.46	0.48	1201
dnschanger	0.49	1.00	0.66	1187
fobber	0.75	0.99	0.86	1203
murofet	0.80	0.89	0.84	1237
necurs	0.90	0.58	0.70	993
newgoz	0.98	0.98	0.98	1185
padcrypt	0.74	0.77	0.76	1145
proslikefan	0.67	0.60	0.63	1176
pykspa	0.49	0.62	0.55	1194
qadars	0.97	0.97	0.97	1172
qakbot	0.50	0.28	0.36	1209
ramdo	0.90	0.98	0.94	1198
ranbyus	0.81	0.76	0.78	1258
simda	0.92	0.97	0.95	1195
suppobox	0.89	0.84	0.86	1234
symmi	0.98	0.99	0.99	1173
tempedreve	0.58	0.39	0.47	1179
tinba	0.87	0.65	0.75	1197
accuracy			0.83	31822
macro avg	0.80	0.79	0.78	31822
weighted avg	0.84	0.83	0.82	31822

TABLE III. RESULTS OF MULTICLASS CLASSIFICATION FOR GRU

	Precision	Recall	F1-score	Support
benign	0.99	0.99	0.99	8079
banjori	1.00	1.00	1.00	1165
corebot	1.00	1.00	1.00	1242
dircrypt	0.71	0.70	0.71	1201
dnschanger	0.90	0.97	0.94	1187
fobber	0.88	0.95	0.91	1203
murofet	0.93	0.94	0.94	1237
necurs	0.96	0.85	0.90	993
newgoz	1.00	1.00	1.00	1185
padcrypt	1.00	0.99	1.00	1145
proslikefan	0.72	0.71	0.72	1176
pykspa	0.68	0.78	0.73	1194
qadars	1.00	1.00	1.00	1172
qakbot	0.70	0.58	0.63	1209
ramdo	1.00	1.00	1.00	1198
ranbyus	0.89	0.88	0.88	1258
simda	1.00	0.99	1.00	1195
suppobox	0.98	0.99	0.99	1234
symmi	1.00	1.00	1.00	1173
tempedreve	0.76	0.70	0.73	1179
tinba	0.92	0.96	0.94	1197
accuracy			0.92	31822
macro avg	0.91	0.90	0.90	31822
weighted avg	0.92	0.92	0.92	31822

Similarly, Table IV represents the experimental results of Long term short memory for multiclass classification of algorithmically generated domain names to classify the domain name into a family of malwares in terms of accuracy, precision, f1-score, recall and Table IV represents the experimental results of gated recurrent unit for multiclass classification of algorithmically generated domain names to classify the domain name into a family of malwares in terms of accuracy, precision, recall, and f1-score. By looking at Table III and Table V proposed model gated recurrent unit has outperformed for multi-class classification in comparison to RNN and LSTM and other deep learning architectures.

TABLE IV. RESULTS OF MULTICLASS CLASSIFICATION FOR LSTM

	Precision	**Recall**	**F1-score**	**Support**
benign	0.99	0.99	0.99	8079
banjori	1.00	1.00	1.00	1165
corebot	1.00	1.00	1.00	1242
dircrypt	0.67	0.68	0.68	1201
dnschanger	0.90	0.95	0.93	1187
fobber	0.87	0.94	0.90	1203
murofet	0.90	0.94	0.92	1237
necurs	0.94	0.85	0.90	993
newgoz	1.00	1.00	1.00	1185
padcrypt	0.99	0.99	1.00	1145
proslikefan	0.81	0.64	0.71	1176
pykspa	0.68	0.80	0.73	1194
qadars	1.00	0.99	0.99	1172
qakbot	0.65	0.54	0.59	1209
ramdo	1.00	1.00	1.00	1198
ranbyus	0.89	0.86	0.87	1258
simda	0.99	1.00	1.00	1195
suppobox	0.98	0.99	0.99	1234
symmi	1.00	1.00	1.00	1173
tempedreve	0.71	0.76	0.73	1179
tinba	0.94	0.93	0.93	1197
accuracy			0.92	31822
macro avg.	0.90	0.90	0.90	31822
weighted avg.	0.92	0.92	0.92	31822

Table V incorporates experimental findings from deep learning methods applied at the character level, alongside logistic regression (LR) using character bigrams, within the specified domain for classification as the domain name into corresponding malware family.

TABLE V. RESULTS OF MULTICLASS CLASSIFICATION FOR BIGRAM WITH LOGISTIC REGRESSION

	Precision	**Recall**	**F1-score**	**Support**
benign	0.98	0.99	0.99	8079
banjori	1.00	1.00	1.00	1165
corebot	1.00	1.00	1.00	1242
dircrypt	0.52	0.57	0.54	1201
dnschanger	0.57	0.85	0.68	1187
fobber	0.71	0.94	0.81	1203
murofet	0.96	0.95	0.96	1237
necurs	0.88	0.71	0.79	993
newgoz	1.00	0.99	1.00	1185
padcrypt	1.00	1.00	1.00	1145
proslikefan	0.78	0.58	0.66	1176
pykspa	0.55	0.58	0.57	1194
qadars	1.00	1.00	1.00	1172
qakbot	0.65	0.47	0.54	1209
ramdo	0.98	1.00	0.99	1198
ranbyus	0.79	0.72	0.75	1258
simda	0.99	0.99	0.99	1195
suppobox	0.97	1.00	0.99	1234
symmi	1.00	1.00	1.00	1173
tempedreve	0.56	0.39	0.46	1179
tinba	0.80	0.84	0.82	1197
accuracy			0.87	31822
macro avg.	0.84	0.84	0.83	31822
weighted avg.	0.87	0.87	0.87	31822

IX. CONCLUSION

In this paper we propose a novel deep learning framework for the detection of malicious domain names, achieving superior performance accuracy for both binary and multiclass classification tasks. Our proposed model uses deep learning techniques with Keras embedding and it has the capability to detect the domain names to a particular malware family. Further the domain names are differentiated as either legitimate or DGA generated by training the domain names within the character level by automatically extracting the necessary features. Detecting DGAs presents a significant challenge in cyber security. These algorithms are typically employed by attackers to establish communication with diverse servers. This paper introduces deep learning architectures designed to mitigate DGA threats. The proposed framework includes a feature extractor and preprocessing model tailored for classifying and detecting malicious domain names. As the volume of data grows, deep learning models offer superior performance compared to traditional machine learning algorithms. This study examines the efficacy of different approaches in detecting DGAs and categorizing domain names into respective families, utilizing a dataset encompassing 20 malware families. Specifically, in the GRU model, domain names are vectorized through a Keras embedding technique, where each domain character is mapped to a vector in a defined dictionary. Future work could explore training with more complex models and adding additional layers for enhanced accuracy. Experimentation with different preprocessing techniques, embeddings, hyper parameter fine-tuning, and increased epochs may further refine the model's accuracy.

REFERENCES

[1] Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C., 2014. EXPOSURE: a passive DNS analysis service to detect and report malicious domains. ACM Trans. Inf. Syst. Secur.16 (4).doi: 10.1145/2584679.

[2] Martin Grill, Ivan Nikolaev, Veronica Valeros, and Martin Rehak. 2015. DetectingDGA Malware Using NetFlow. In IFIP/IEEE International Symposium on IntegratedNetwork Management.

[3] Manos Antonakakis, Roberto Perdisci, YacinNadji, NikolaosVasiloglou, SaeedAbu-Nimeh, Wenke Lee, and David Dagon. 2012. From Throw-Away Traffic toBots: Detecting the Rise of DGA-Based Malware. In USENIX Security Symposium.

[4] Daniel Plohmann, Fraunhofer and KhaledYakdan A Comprehensive Measurement Study of Domain Generating Malware.25th USENIX Security Symposium August 10–12, 2016, Austin, TX ISBN 978-1-931971-32-4.

[5] Samuel Schüppen, DominikTeubert, Patrick Herrmann, and Ulrike Meyer. 2018. FANCI : Feature-based Automated NXDomain Classification and Intelligence. In 27th USENIX Security Symposium (USENIX Security 18). USENIX Association, Baltimore, MD, 1165–1181.

[6] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan. 2012. Detecting Algorithmically Generated Domain-Flux Attacks with DNS Traffic Analysis.IEEE/ACM Transactions on Networking 20, 5 (Oct 2012), 1663–1677. https: //doi.org/10.1109/TNET.2012.2184552.

[7] D. T. Truong and G. Cheng, Detecting domain-flux botnet based on DNS traffic features in managed network, Secur.Commun. Networks, vol. 9, no. 14, 2016, pp. 2338–2347.

[8] Pereira M, Coleman S, Yu B, DeCock M,Nascimento A (2018) Dictionary extraction and detection of algorithmically generated domain names in passive dns traffic. In: International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, Heraklion.pp 295–314.

[9] K. Alieyan, A. ALmomani, A. Manasrah, M. M. J. N. C. Kadhum, and Applications, "A survey of botnet detection based on DNS," vol. 28, no. 7, pp. 1541-1558, 2017.

[10] V. Tong and G. Nguyen, A method for detecting DGA botnet based on semantic and cluster analysis, inProc. Seventh Symp.on Information and Communication Technology, Ho Chi Minh City, Vietnam, 2016, pp. 272–277.

[11] Kheir, M., Rossow, C., &Holz, T. (2014, September). Paint it black: Evaluating the effectiveness of malware blacklists. In International Workshop on Recent Advances in Intrusion Detection (pp. 1-21). Springer, Cham.

[12] Jonathan Woodbridge, Hyrum S. Anderson, AnjumAhuja, and Daniel Grant. 2016. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. CoRRabs/1611.00791 (2016).arXiv:1611.00791 http://arxiv.org/abs/ 1611.00791.

[13] Shibahara T, Yagi T, Akiyama M, Chiba D, Yada T (2016) Efficient dynamic malware analysis based on network behavior using deep learning. In: 2016 IEEE Global Communications Conference (GLOBECOM). IEEE, Washington, DC.pp 1–7.

[14] Geffner J (2013) End-to-end analysis of a domain generating algorithm malware family. In: Black Hat USA 2013.

[15] F. Zeng, S. Chang, and X. C. Wan, Classification forDGA-based malicious domain names with deep learningarchitectures, Int. J. Intell. Inf. Syst., vol. 6, no. 6, pp. 67–71,2017.

[16] 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.

[17] Qiao, Y.; Zhang, B.; Zhang, W.; Sangaiah, A.K.; Wu, H. DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism. Appl. Sci. 2019, 9, 4205.

[18] Y. C. Cheng, Y. J. Li, A. Tseng, and T. Lin, Deep learning for malicious flow detection, arXiv preprint arXiv: 1802.03358, 2018.

[19] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, Predicting domain generation algorithms with long shortterm memory networks, arXiv preprint arXiv: 1611.00791, 2016.

[20] Y. Li, K. Q. Xiong, T. Chin, and C. Hu, A machine learning framework for domain generation algorithm-based malware detection, IEEE Access, vol. 7, pp. 32 765–32 782, 2019.

[21] C. D. Chang and H. T. Lin, On similarities of string and query sequence for DGA botnet detection, in Proc. 2018 Int. Conf. on Information Networking, Chiang Mai, Thailand, 2018, pp. 104–109.

[22] Yang L, Liu G, Zhai J, Dai Y, Yan Z, Zou Y, Huang W (2018) A novel detection method for word-based dga. In: International Conference on Cloud Computing and Security. Springer, Haikou.pp 472–483.

[23] Zang X, J G, X H (2018) Detecting malicious domain name based on agd. J Commun 39(7):15–25.

[24] Yu, Bin, et al. "Character level-based detection of DGA domain names." 2018 International Joint Conference on Neural Networks (IJCNN).IEEE, 2018.

[25] Akarsh, S., et al. "Deep learning framework for domain generation algorithms prediction using long short-term memory." 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS).IEEE, 2019.

[26] Lison, P.; Mavroeidis, V. Automatic Detection of Malware-Generated Domains with Recurrent Neural Models. arXiv 2017, arXiv:1709.07102.

[27] Mac, H.; Tran, D.; Tong, V.; Nguyen, L.G.; Tran, H.A. DGA Botnet Detection Using Supervised Learning Methods. In Proceedings of the 8th International Symposium on Information and Communication Technology, Nhatrang, Vietnam, 7–8 December 2017; pp. 211–218.

[28] Yu, B.; Gray, D.L.; Pan, J.; de Cock, M.; Nascimento, A.C.A. Inline DGA detection with deep networks. In Proceedings of the 2017 IEEE International Conference Data Mining Workshops (ICDMW), New Orleans, LA, USA, 18–21 November 2017; pp. 683–692.

[29] Zeng, F.; Chang, S.; Wan, X. Classification for DGA-Based Malicious Domain Names with Deep Learning Architectures. Int. J. Intell. Inf. Syst. 2017, 6, 67–71.

[30] Lison, P., &Mavroeidis, V. (2017). Automatic Detection of Malware Generated Domains with Recurrent Neural Models. arXiv preprint arXiv:1709.07102.

[31] Ryan R. Curtin, Andrew B. Gardner and SlawomirGrzonkowski “Detecting DGA domains with recurrent neural networks and side information.”ARES ’19, August 26–29, 2019, Canterbury, NY.

[32] Anderson, H.S.; Woodbridge, J.; Filar, B. DeepDGA: Adversarially-tuned domain generation and detection. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, Vienna, Austria, 28 October 2016; pp. 13–21.

[33] J Koh, Rhodes B. Inline detection of domain generation algorithms with context-sensitive word embeddings. 2018.

[34] Chin, T.; Xiong, K.Q.; Hu, C.B.; Li, Y. A machine learning framework for studying domain generation algorithm (DGA)-based malware.In Proceedings of the International Conference on Security and Privacy in Communication Systems, Singapore, 8–10 August 2018.

[35] Vinayakumar, R., Poornachandran, P., &Soman, K. P. (2018). Scalable Framework for Cyber Threat Situational Awareness Based on Domain Name Systems Data Analysis. In Big Data in Engineering Applications (pp. 113-142).Springer, Singapore.

[36] Vinayakumar, R., Soman, K. P., &Poornachandran, P. (2018). Detecting malicious domain names using deep learning approaches at scale. Journal of Intelligent & Fuzzy Systems, 34(3), 1355-1367.