ShadowDGA: Toward Evading DGA Detectors with GANs

Yu Zheng, Chao Yang*, Yanzhou Yang, Qixian Ren, Yue Li, Jianfeng Ma
School of Cyber Engineering, Xidian University, Xi'An, China
Email: yuzheng.xidian@gmail.com, chaoyang@xidian.edu.cn, 1163839161@qq.com,
gson307@outlook.com, liyue@xidian.edu.cn, jfma@mail.xidian.edu.cn

2021 International Conference on Computer Communications and Networks (ICCCN) | 978-1-6654-1278-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCN52240.2021.9522282

***Abstract*—Domain generation algorithms (DGAs) are widely used in modern botnets to generate a large number of domain names through which bots can communicate with their command and control (C&C) servers. In recent years, many machine learning based approaches have been proposed to automatically detect algorithmically generated domains in real time and have achieved success in traditional DGAs. Nevertheless, they are somewhat unavailable for adversarial domains. In this paper, we develop a more threatening DGA called ShadowDGA that utilizes generative adversarial networks (GANs) to simulate the distribution of benign domains without any knowledge about the DGA detector to evade detection. Experimental results demonstrate that the domains generated by ShadowDGA are the most difficult to detect compared to existing DGA families. We also present an effective defense method for adversarial domains without retraining. These findings indicate that detectors that rely solely on features extracted from the domain name are vulnerable, while a robust DGA detector should contain additional contextual information.**

***Index Terms*—DGA Detection, Adversarial Learning, GAN, C&C Server**

I. Introduction

Botnets are collections of malware-infected machines (bots) that are remotely controlled by an attacker (botmaster) through a command and control (C&C) communication channel [1]. Since hardcoded IP addresses in malware binary are easily blacklisted, botmasters have already started employing domain generation algorithms (DGAs) to strengthen the C&C infrastructure of their botnets. DGAs are used to dynamically generate a large number of domain names computed with a publicly available seed, e.g., the current date, Twitter trends, and weather forecasts. Compromised computers attempt to query these generated domain names, one of which is registered by the botmaster, to receive updates or commands. Once such a query is successfully resolved, communication between bots and the C&C server occurs. However, law enforcement and security researchers must block access to all of the domains. The use of DGAs makes it more difficult to shut down botnets.

In the past decade, a multitude of studies has been devoted to detect DGA domain names by employing machine learning technologies. The approaches can be broadly divided into two categories: "featureful" and "featureless" [2]. The "featureful" approaches rely on features manually extracted from the domain names. Yadav et al. first divides the domain names into different groups, then computes the different metrics over them, and finally uses L1 normalized linear regression to batch detect DGA domain names [3]. Schuppen et al. extracted 21 features in 3 categories from the individual nonexistent domain to detect algorithmically generated domains with random forest (RF) [4]. The "featureless" approaches automatically derive features for detection. Popular kinds of detection models used in the featureful approaches, including the long short-term memory (LSTM) model [5], the convolutional neural networks (CNN) model [6], [7], and the CNN and LSTM hybrid model [6], [8] are deep neural network-based models and yield better effects.

Nevertheless, machine learning models are powerless against adversarial examples that an attacker has intentionally designed to cause the model to make a mistake [9]. In the context of DGA detection, DeepDGA [10] exploits a generative model to produce domains that are difficult to detect. DeepDGA domains can also be used to expand the training set to enhance the generalization capabilities of the detection model. However, DeepDGA has a more complicated structure and is difficult to train, which is exactly the problem that our study solves. MaskDGA [11] adds perturbation to existing algorithmically generated domains to evade DGA detectors without any knowledge of the targeted model. The perturbation is so subtle that the detection effect does not decrease too much. Peck et al. proposed CharBot[12], which replaces two characters in a benign domain name with other valid characters. The domains generated in this way are likely to be premium domains, increasing the attacker's costs. In addition, Spooren et al. leverages the knowledge of the features to design a new DGA that makes the random forest classifier powerless [13], but it performs poorly when evading deep learning models that automatically extract potential features of domain names.

In this paper, we design a practical and more menacing domain generation algorithm called ShadowDGA. ShadowDGA takes advantage of generative adversarial networks to simulate the distribution of benign domain names. We build a generator and discriminator with one-dimensional convolutional neural networks that require fewer parameters, resulting in less training time. The residual block is employed to reduce the error rate of model, and the gradient penalty [14] is

*corresponding author

applied to the loss function of discriminator to solve the mode collapse problem. The domains generated by ShadowDGA are almost unregistered and have various patterns. Moreover, the model file is smaller than MaskDGA. ShaowDGA is subsequently evaluated on four recently proposed detectors. The experimental results show that the ShadowDGA domains can achieve best results among all measures and are the hardest to detect. Finally, we also apply a method that leverages an autoencoder to reconstruct inputs to defend against adversarial domains without retraining.

Our main contributions are summarized as follows:

- We propose a practical and more threatening DGA to generate adversarial domains.
- Experimental results demonstrate that the proposed DGA has better escape performance than the existing DGA families on four recently developed DGA detectors.
- We present a more general approach to adversarial domain defense without retraining.
- These findings show that DGA detectors should not only rely on domain name features but also contain contextual information.

II. Related Work

At present, many machine learning methods have been applied in the field of DGA detection, such as: random forest [15], support vector machine [16] and unsupervised learning model [17]. However, most DGA detection methods based on traditional machine learning rely on hand-crafted features [18], [19]. The disadvantages are twofold: First, not all features are designed for the specific task. Therefore, some features have no effect on the machine learning model, or even have a counter effect. Second, the attacker can adjust DGA according to the characteristics of these hand-crafted features to avoid detection.

Therefore, many DGA detection techniques based on deep learning have been proposed to automatically detect algorithmically generated domains in real time. Woodbridge et al. used a single-layer LSTM to complete the DGA detection task for the first time [5]. Further, Lison et al. used the bidirectional LSTM to improve the detection performance [20]. Some scholars introduce the CNN model in text classification into DGA detection [7], [21]. Other scholars combine the CNN model and the LSTM model, employing their respective advantages to deal with this problem [8], [22], [23].

However, machine learning models, including neural networks, are vulnerable to crafted adversarial examples [24]. DGA detectors that extract features only from the domain name are also easily fooled by adversarial domains.

Anderson et al. presented DeepDGA [10], which leverages a generative model to pseudorandomly produce domain names that are difficult for modern DGA classifiers to detect. They also find that a classifier can be strengthened by augmenting the training set with DeepDGA domains. They pretrain an autoencoder to represent valid domain names and then repurpose the encoder as a discriminative model and the decoder as a generative model. DeepDGA employs LSTM as its internal structure, resulting in many parameter calculations. In addition, the domain names generated by DeepDGA do not have rich patterns due to the problem of mode collapse.

Sidi et al. proposed MaskDGA [11], a black-box technique for evading DGA detectors, without prior knowledge of the detector architecture and parameters. It is conducted by training a substitute model on public datasets first. The attacker then adds perturbation to the character-level representation of DGA domains based on the Jacobian saliency map to evade detection. The detection rate of adversarial domains generated by MaskDGA is relatively high because the perturbation is subtle.

Peck et al. developed a novel DGA called CharBot that can produce large numbers of adversarial domains by replacing two characters in the benign domain name with characters chosen from an equal distribution of DNS-valid characters [12]. Although they claim that CharBot domains are unregistered, the homomorphic domain names obtained through character replacement are more likely to be premium domains, leading to high attack costs.

DeceptionDGA [13], a new DGA developed by Spooren et al., considers manually engineered features to circumvent detection. This DGA makes the random forest detector powerless with an accuracy of 59.9%. However, DeceptionDGA needs to obtain knowledge about the features used by a DGA detector and is inefficient for deep learning-based detectors

III. Threat Model

As shown in Figure 1, there are three main components in our threat model: defender, attacker and datasets. Note that the datasets are public datasets.

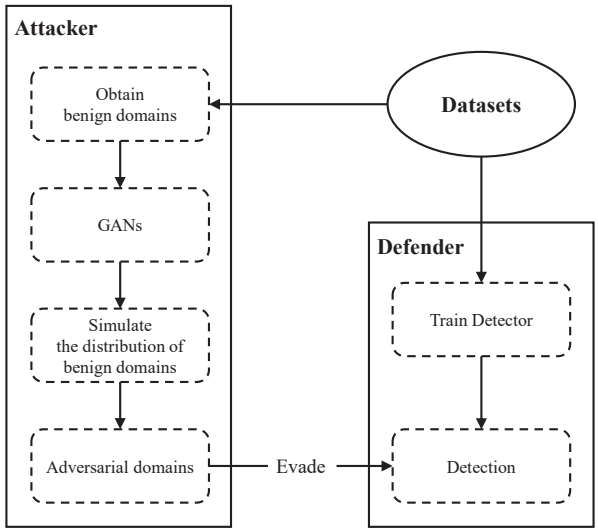


Figure 1. Threat Model

The detector is trained with public datasets and deployed to the network by the defender. It monitors all the network traffic between hosts and the DNS server and checks if there are any machines attempting to query DGA domains. We assume

that the attacker cannot access the detector and has no idea about the architecture of the targeted model. Nevertheless, the attacker can obtain benign domains from public datasets and then leverage GANs to simulate the distribution of benign domains. The adversarial domains generated in this way are difficult to identify by the detector. This is a new threat that requires more attention from the security community.

IV. SHADOWDGA

GAN is a kind of machine learning method that learns by letting two neural networks play against each other. Given training data, this technique learns to generate new data with the same statistics as the training data. In the past few years, GANs have achieved great success in the field of computer vision and natural language processing [25]. Inspired by this, we designed a more threatening domain generation algorithm that utilizes GANs to simulate benign domains to generate adversarial domains. In this section, we first describe the implementation steps and specific structure of ShadowDGA, then introduce the experimental setup, and finally show the samples of adversarial domains generated by ShadowDGA.

A. Method

ShadowDGA is a GAN-based domain generation algorithm intended to evade DGA detectors. Similar to other works, we only focus on the second-level domain (SLD) and the top-level domain (TLD) throughout this paper. The whole process of implementing ShadowDGA is shown in Algorithm 1. We (1) take SLDs M and TLDs T as input to ShadowDGA, (2) train a GAN that includes a generator and a discriminator, (3) generate a set of SLDs S with an optimal generator, (4) add a TLD t randomly selected from T for each SLD s in S, and finally (5) obtain adversarial domains that can fool DGA detectors. The maximum number of iterations for training the GAN model is N. In each iteration, the discriminator is updated K times, and the generator is updated one time. The loss function of the generator L_D and the loss function of the discriminator are the same as in [14].

The discriminator takes prior samples from the output of the generator and SLDs from real data as its input and feeds them into a one-dimensional convolution (Conv1D) layer. The output of this layer is then fed into five sequentially connected residual blocks. The output of the last residual block is fed into a fully connected layer.

The generator is loosely the reverse of the discriminator. It takes noise samples from a random normal distribution as its input. The noise samples are fed into a fully connected layer. The output of this layer is then fed into five sequentially connected residual blocks. The output of the last residual block is fed into a Conv1D layer and a softmax activation function. In addition, the output of this layer is the generated SLD.

The residual block shown in Figure 2 is loosely inspired by deep residual learning [26]. Assume x is the input of a neural network and $H(x)$ is the expected output. $H(x) = x$ is the expected complex potential mapping, and learning is difficult. Hence, the residual block directly passes the input x to the output as the initial result through shortcut connections. It no longer learns a complete output but the difference between the optimal solution $H(x)$ and the identity mapping x; that is, the residual $F(x) = H(x) - x$. The residual block can effectively reduce the error rate of the network so that adversarial domains can better simulate the distribution of benign domains through the generative adversarial network.

Algorithm 1 ShadowDGA

Input: A set of SLDs M, A set of TLDs T
Parameter: Maximum number of iterations N, number of discriminator updates in each iteration K
Output: adversarial domains

Let $n = 0$, $k = 0$
while $n < N$ **do**
  if $k < K$ **then**
    Sample minibatch of m prior samples.
    Sample minibatch of m SLDs from M.
    Update the discriminators weights θ_d by descending along the gradient $\nabla_{\theta_d} L_D$
    $k = k + 1$
  end if
  Sample minibatch of m noise samples.
  Update the generators weights θ_g by descending along the gradient:$\nabla_{\theta_g} L_G$
  $n = n + 1$
end while
Generate a set of SLDs S with the optimal generator.
Let $R = \emptyset$
while s in S **do**
  Randomly select a TLD t from T.
  Add $s.t$ to R.
end while
return R

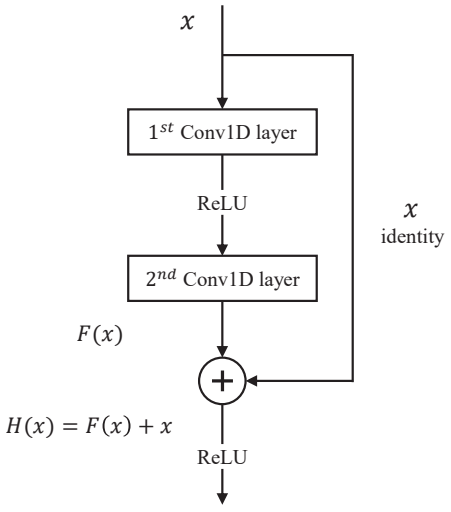


Figure 2. Residual block

B. Experimental Setup

In this experiment, we employ tldextract[1] to extract SLDs of all one million domain names in the Tranco list. The TLDs we used are composed of *com*, *net*, *org*, *info*, *biz*, *ru*, *in* and *cc*, which are the seven most popular TLDs used by DGAs [27]. We utilize TensorFlow to build the GAN model. The maximum number of iterations is set to 20,000, and the number of discriminator updates in each iteration is set to 10. The model is trained with a batch size of 64 until convergence using the ADAM optimizer with a learning rate of 0.0001.

C. Results Analysis

We monitor JensenShannon (JS) divergence between the real bigram distributions and the generated bigram distributions. The change in the JS distance with the number of iterations is shown in Figure 3. We find that there is almost no additional increase in the JS distance after 5,000 iterations. This shows that the domains generated by ShadowDGA can well simulate the distribution of benign domains. Therefore, we choose the generator obtained from 5,000 iterations as the optimal generator to generate adversarial domains. In addition, we calculated the number of bigrams in 10,000 ShadowDGA domain names and 10,000 DeepDGA domain names. There are 872 bigrams in ShadowDGA and only 545 bigrams in DeepDGA, showing that adversarial domains generated by ShadowDGA have richer models.

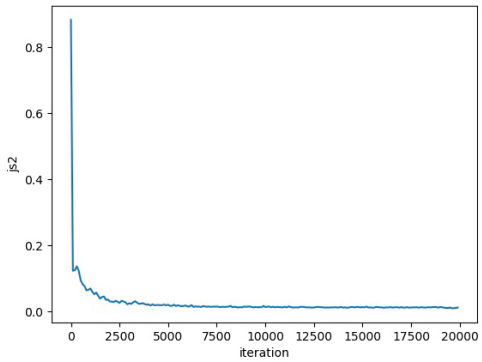


Figure 3. Changes in JS divergence with the number of iterations

Table I shows a few adversarial domains generated by ShadowDGA. These domains vary in length, have no fixed pattern and look like benign domains at first glance. A convolutional neural network (CNN) is a feedforward neural network with a parameter sharing scheme. We only use one-dimensional convolutional neural networks to build generators and discriminators, avoiding a large number of parameter calculations. A total of 5,000 iterations of training took 37 minutes on a single NVIDIA GeForce GTX TITAN X GPU, far less than the 14 hours of the DeepDGA pretraining phase. This shows that the model we proposed is easier to train.

[1]https://github.com/john-kurkowski/tldextract

Table I
EXAMPLE OF SHADOWDGA DOMAINS

vankespn.com	manfunkp.info
cereranany.cc	vinbagger.info
herasrootk.net	muskwordth.ru
haoestorky.biz	lhyoncioprip.com
cpargp.cc	fansrikurstoamis.in
malharohy.biz	sthalycards.org
vinbagger.info	fridekyvecirus.com
kustparks.com	

Finally, we analyze the practicality of ShadowDGA. A domain generation algorithm is unsuccessful if most of the domain names it generates have already been registered. We randomly selected 500 domain names and then went to the registration website to check, but none of them were registered. The ShadowDGA model that we trained can be serialized to an 8.5M HDF5 file, which is 1.5M smaller than the substitute model in MaskDGA. Therefore, we do not think file size is an obstacle when deploying ShadowDGA.

V. EVASION

We conduct experiments on four recently proposed DGA detectors (FANCI [4], Endgame [5], NYU and MIT [6]) to evaluate the detectability of ShadowDGA. All detectors are trained to mark a given domain name as benign or DGA. We also compare ShadowDGA with several traditional DGA families and DeepDGA.

A. Datasets

We first introduce the datasets used in this experiment. The ground truth (GT) datasets for both benign and DGA domain names are taken from public locations for reproducibility.

Benign Ground Truth. Many previous works chose Alexa top sites[2] as benign domain names. However, references [28] and [29] showed that the Alexa list is easily manipulated by an adversary and contains malicious domain names. Thus, we use the Tranco list[3] created on 26 November 2019 as the GT for benign domain names in this work. Tranco is a new ranking provided by [29] to improve upon the shortcomings of current lists. We orderly select 110,000 domain names, of which the first 100,000 are used for training and the remaining 10,000 are used for testing.

DGA Ground Truth. Thirteen traditional DGAs are selected from the DGArchive [27], which is a free service offered by Fraunhofer FKIE and administered by Daniel Plohmann. Additionally, we reimplemented DeepDGA with the source code disclosed by Hyrum Anderson. Hence, including the ShadowDGA proposed in this paper, there are fifteen DGA families (see Table II) with 10,000 samples in each family. They can be divided into four types according to the generation schemes:

[2]https://www.alexa.com/topsites
[3]https://tranco-list.eu/list/7N2X

Table II
INFORMATION ABOUT DGAS

Family	Type	L_{min}	L_{max}	Purpose
Banjori	A	7	26	$*$
Conficker	A	5	11	$*$
Corebot	A	12	28	$*$
Dyre	H	34	34	$*$
Matsnu	W	12	24	$*$
Necurs	A	7	25	$*$
Pykspa	A	6	12	$*$
QakBot	A	8	25	$*$
ROVNIX	A	18	18	$*$
SimDa	A	5	11	$*$
Bamital	H	32	35	†
DeepDGA	DL	5	12	†
Gozi	W	12	24	†
Ramnit	A	8	19	†
ShadowDGA	DL	6	16	†

- Arithmetic-based: calculating a sequence of alphanumeric characters.
- Hash-based: employing the hexadecimal representation of a hash value.
- Wordlist-based: concatenating a sequence of words.
- Deep learning-based: generating DGA domains using deep learning techniques.

Finally, we choose five DGA families (Bamital, DeepDGA, Gozi, Gamnit and ShaowDGA) to evaluate their detectability. The remaining ten DGA families are used as positive samples in the training set.

B. Detectors

The target models that DGA families try to evade are based on the following recently proposed DGA detectors:

- **FANCI.** [4] presented a random forest classifier for detecting DGA domains by monitoring non-existent domain responses in DNS traffic. It uses 21 features that can be extracted from an individual domain name. These features can be grouped into three categories: structural features, linguistic features and statistical features. As we only consider second-level domain, a number of features used in the FANCI don't work in our experiments. Table III lists the features we used.
- **Endgame.** [5] used deep learning for DGA detection for the first time. The Endgame model is a neural network for binary classification consisting of following sequential layers: an embedding layer, an LSTM layer and a single node output layer with sigmoid activation.
- **NYU.** To the best of our knowledge, [7] is the first to successfully apply character-level convolutional networks on text classification. The model includes six stacked CNN layers, with each previous layer feeding output to the next layer. Domain names are too short and have no specific semantics, which may cause the above model to overfit. [6] therefore reduced the number of stacked CNN layers to two for DGA detection.

Table III
FANCI FEATURES USED IN THE EVALUATION

#	Feature
1	Domain Name Length
2	Underscore Ratio
3	Contains IP Address
4	Contains Digits
5	Vowel Ratio
6	Digit Ratio
7	Alphabet Cardinality
8	Ratio of Repeated Characters
9	Ratio of Consecutive Consonants
10	Ratio of Consecutive Digits
11	N-Gram Dist
12	Entropy

- **MIT.** The MIT model proposed by [8] consists of stacked CNN layers and single LSTM layer. For the same reason with NYU [6] reduced the MIT model to the minimum: one CNN layer followed by one LSTM layer.

The FANCI model is implemented using scikit-learn[30] and based on the code published on a github repository[4]. Source codes for other detectors are available on [6].

C. Performance

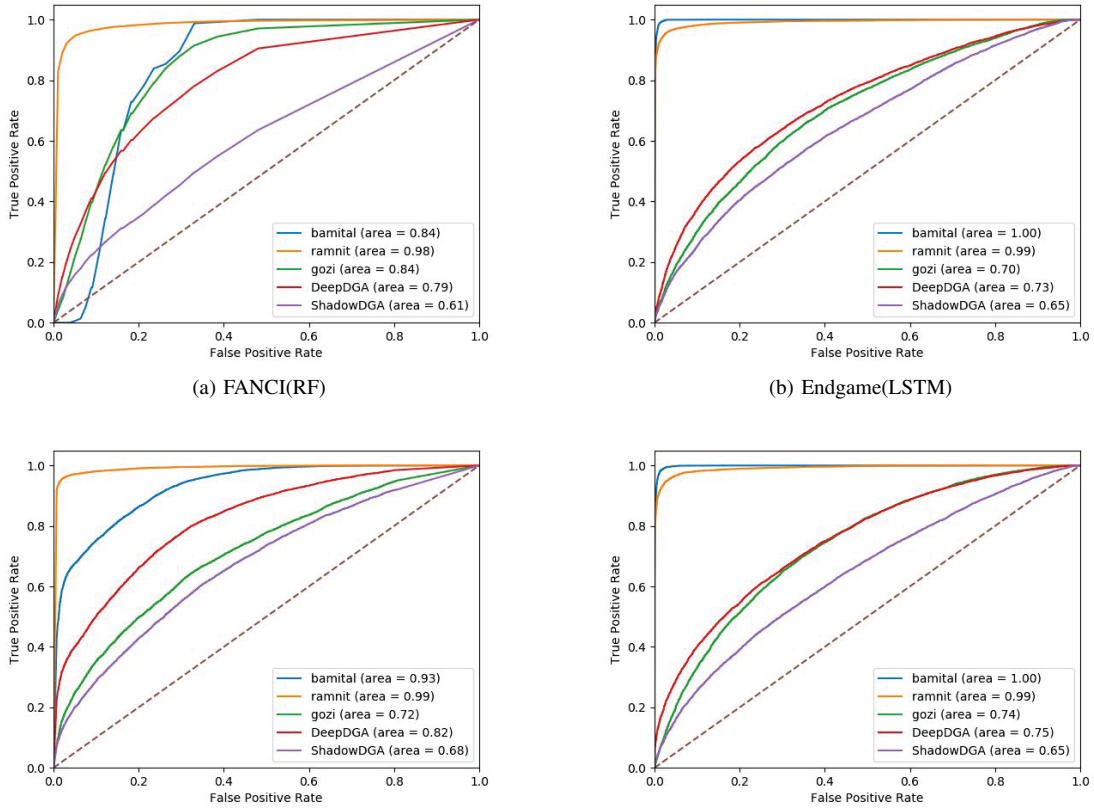
Table II shows some information about DGAs. There are four types of DGAs (A: arithmetic-based; H: hash-based; W: wordlist-based; DL: deep learning-based). L_{min} is the minimum domain length. L_{max} is the maximum domain length. $*$ represents DGA for training. † represents DGA for testing. The detectors are trained on 100k DGA domains as malicious and the Tranco top 100k domains as benign. The DGA domains contain ten families with 10k domains in each family. Then, we use the trained model to detect five DGA families individually that have not appeared in the training set. Each testing set contains 10K DGA domains and 10K benign domains.

In this experiment, we will use four commonly used metrics (accuracy, precision, recall and F1-score) to measure the results of the experiment. In the binary classification problem, when the classifier prediction is completed, four classification results can be obtained: 1) True Positive (TP): successfully predict positive samples as positive; 2) True Negative (TN): successfully predict negative samples as negative; 3) False Positive (FP): falsely predict negative samples as positive; 4) False Negative (FN): falsely predict positive samples as negative. The four metrics used in this experiment are based on these four classification results, defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

[4]https://github.com/fanci-dga-detection/fanci



(a) FANCI(RF)

(b) Endgame(LSTM)

(c) NYU(CNN)

(d) MIT(LSTM+CNN)

Figure 4. Receiver Operating Characteristic curve

Table IV
ACCURACY, PRECISION, RECALL AND F1-SCORE

Detector	Metric	Bamital	Ramnit	Fozi	DeepDGA	ShadowDGA
FANCI(RF)	Accuracy	0.61	0.92	0.70	0.69	**0.57**
	Precision	0.74	0.89	0.81	0.80	**0.69**
	Recall	0.34	0.97	0.52	0.49	**0.27**
	F1-score	0.47	0.93	0.63	0.61	**0.38**
Endgame(LSTM)	Accuracy	0.97	0.96	0.57	0.60	**0.56**
	Precision	0.95	0.95	0.79	0.83	**0.77**
	Recall	1.00	0.97	0.19	0.25	**0.17**
	F1-score	0.97	0.96	0.31	0.39	**0.28**
NYU(CNN)	Accuracy	0.82	0.96	0.60	0.68	**0.58**
	Precision	0.92	0.94	0.82	0.87	**0.78**
	Recall	0.69	0.97	0.27	0.42	**0.22**
	F1-score	0.79	0.96	0.40	0.57	**0.34**
MIT(CNN-LSTM)	Accuracy	0.97	0.96	0.58	0.63	**0.56**
	Precision	0.94	0.94	0.79	0.84	**0.75**
	Recall	1.00	0.97	0.23	0.31	**0.18**
	F1-score	0.97	0.96	0.35	0.45	**0.29**

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Table IV presents the experimental results when using the test DGA family to evade the four target detectors. These detectors cover a variety of typical machine learning algorithms and can represent the latest DGA domain detection technology. We compare the performance of ShadowDGA with four other DGA families (Bamital, Gozi, Ramnit and DeepDGA).

Based on the results, we can see that all detectors perform almost perfectly on Ramnit, one of the arithmetic-based DGAs that are the most common type of DGA. In addition, More specifically, the targeted detectors yield an average accuracy (0.95), precision (0.93), recall (0.97), and F1-score (0.95), confirming that the recently proposed DGA detection methods are effective for general DGA families. However, there are only 0.57 accuracy, 0.75 precision, 0.21 recall and 0.32 F1-score on average for ShadowDGA. These metrics are significantly smaller than those of the traditional DGA families, indicating that our proposed DGA is not easy to detect.

We do our best to find all DGA domains to block C&C communication. Reference [31] claims that Gozi is a DGA family that resembles English words and is difficult to detect. The results in Table IV show that an average of 21% of ShaowDGA domains is detected by these detectors (recall), whereas the next best DGA, gozi, is detected at an average rate of 30%. This represents a 9% improvement in evading detection. As opposed to DeepDGA, the first deep learning-based DGA family, detectors also show worse performance on ShadowDGA with a recall decrease from 37% to 21%.

Receiver operating characteristic curves (ROC curves) are

typically used in binary classification to intuitively reflect the performance of a classifier. Figure 4 shows ROC curves for detectors on different testing sets. In the four subfigures, the purple curve representing ShadowDGA is always at the bottom. In the figure, we can observe that our designed ShadowDGA outperforms each of the other DGAs, providing better performance in avoiding detection. From the area under the curve (AUC), we also find that the deep learning models generally achieve better detection results than the random forest model.

VI. Defenses

Adversarial samples that can lead the model to misclassify widely exist in various machine learning tasks. The vulnerability to adversarial examples becomes one of the major risks for applying machine learning in safety-critical environments [32]. Adversarial training is considered an effective defense method for adversarial samples. This method augments the training set with adversarial samples to retrain the model. However, it requires prior knowledge about adversarial samples and lacks generality.

Meng et al. presented a new adversarial defense framework called MagNet that neither modifies the protected classifier nor requires prior knowledge [33]. MagNet first moves adversarial examples toward the manifold of normal examples with an autoencoder and then feeds the reformed samples into the classifier, resulting in improved classification accuracy of the model. We apply MagNet to defend against adversarial domains by changing the structure of the autoencoder. According to the data flow, the encoder consists of one embedding layer, three parallel Conv1D layers with a ReLU activation function, one concatenate layer, and one Conv1D layer with a ReLU activation function. In addition, the decoder consists of three parallel Conv1D layers with a ReLU activation function, one concatenate layer, one Conv1D layer with a ReLU activation function and one fully connected layer with a softmax activation function. The autoencoder is trained with the same training set as the DGA detectors.

We tested the defense effectiveness of adversarial training and MagNet on the MIT model combining LSTM and CNN. As shown in Table V, the detection accuracy of adversarial domains used to enhance the training set has been greatly improved, but the detection accuracy of other adversarial domains has hardly improved. MagNet improves the detection accuracy of all adversarial domains by an average of 20%, which indicates that it is more general.

Table V
Defense effectiveness of adversarial training and MagNet

	DeepDGA	ShadowDGA
No Defense	0.63	0.56
DeepDGA Retraining	0.92	0.57
ShadowDGA Retraining	0.66	0.86
MagNet	0.81	0.78

VII. Conclusion

In this paper, we proposed ShadowDGA, a more threatening domain generation algorithm. ShadowDGA takes advantage of GANs to simulate the distribution of benign domain names. The experimental results on four DGA detectors using different machine learning algorithms show that ShadowDGA domains have better evasive performance compared to other DGAs. Simultaneously, we also presented a more general approach to defend against adversarial domains. The results demonstrate that DGA detectors based only on the domain name can be easily fooled. Therefore, we should develop robust DGA detectors with context information that is more difficult to manipulate.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions. This work was supported in part by the National Natural Science Foundation of China under Grant 61906143 and 62036007, in part by the Fundamental Research Funds for the Central Universities under Grant XJS211504, in part by the National Key Research and Development Program of China under Grant 2017YFB0801805, in part by the Key Research and Development Program of Shaanxi Province under Grant 2018ZDCXL-G-9-5, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2020JQ-305.

References

[1] H. R. Zeidanloo and A. B. A. Manaf, "Botnet detection by monitoring similar communication patterns," *arXiv preprint arXiv:1004.1232*, 2010.

[2] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. De Cock, "An evaluation of dga classifiers," in *IEEE International Conference on Big Data (BigData)*, 2018, pp. 5058–5067.

[3] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *ACM SIGCOMM conference on Internet measurement (IMC)*, 2010, pp. 48–61.

[4] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "Fanci: Feature-based automated nxdomain classification and intelligence," in *USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1165–1181.

[5] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," *arXiv preprint arXiv:1611.00791*, 2016.

[6] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of dga domain names," in *International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[7] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 649–657.

[8] S. Vosoughi, P. Vijayaraghavan, and D. Roy, "Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder," in *ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR)*, 2016, pp. 1041–1044.

[9] I. Goodfellow, N. Papernot, S. Huang, Y. Duan, P. Abbeel, and J. Clark, "Attacking machine learning with adversarial examples," *OpenAI. https://blog. openai. com/adversarial-example-research*, 2017.

[10] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2016, pp. 13–21.

[11] L. Sidi, A. Nadler, and A. Shabtai, "Maskdga: A black-box evasion technique against dga classifiers and adversarial defenses," *arXiv preprint arXiv:1902.08909*, 2019.

[12] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, A. Nascimento, and M. De Cock, "Charbot: A simple and effective method for evading dga classifiers," *arXiv preprint arXiv:1905.01078*, 2019.

[13] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, and W. Joosen, "Detection of algorithmically generated domain names used by botnets: a dual arms race," in *ACM/SIGAPP Symposium on Applied Computing (SAC)*, 2019, pp. 1916–1923.
[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5767–5777.
[15] A. Ahluwalia, I. Traore, K. Ganame, and N. Agarwal, "Detecting broad length algorithmically generated domains," in *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments (ISDDC)*, 2017, pp. 19–34.
[16] Y. Chen, S. Yan, T. Pang, and R. Chen, "Detection of dga domains based on support vector machine," in *International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, 2018, pp. 1–4.
[17] J. Raghuram, D. J. Miller, and G. Kesidis, "Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling," *Journal of advanced research*, vol. 5, no. 4, pp. 423–433, 2014.
[18] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: Dga-based botnet tracking and intelligence," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2014, pp. 192–211.
[19] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: Detecting the rise of dga-based malware," in *USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 491–506.
[20] P. Lison and V. Mavroeidis, "Automatic detection of malware-generated domains with recurrent neural models," *arXiv preprint arXiv:1709.07102*, 2017.
[21] S. Zhou, L. Lin, J. Yuan, F. Wang, Z. Ling, and J. Cui, "Cnn-based dga detection with high coverage," in *IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2019, pp. 62–67.
[22] G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria, "Ensemble application of convolutional and recurrent neural networks for multi-label text categorization," in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2377–2383.
[23] V. S. Mohan, R. Vinayakumar, K. Soman, and P. Poornachandran, "Spoof net: syntactic patterns for identification of ominous online factors," in *IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 258–263.
[24] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[25] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.
[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
[27] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 263–278.
[28] P. Royal, "Quantifying maliciousness in alexa top-ranked domains," in *Proc. BlackHat*, 2012.
[29] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *The Network and Distributed System Security Symposium (NDSS)*, 2019.
[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[31] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting dga domains with recurrent neural networks and side information," in *International Conference on Availability, Reliability and Security (ARES)*, 2019, p. 20.
[32] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.
[33] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 135–147.