EAI Endorsed Transactions
on Security and Safety
Research Article EAI.EU

CLETer: A Character-level Evasion Technique Against Deep Learning DGA Classifiers

Wanping Liu[1], Zhoulan Zhang[2], Cheng Huang[1], and Yong Fang[1,*]

[1]School of Cyber Science and Engineering, Sichuan University, Chengdu, China
[2]Information security institute, Sichuan University, Chengdu, China

Abstract

The detection of pseudo-random domain names generated by Domain Generation Algorithms (DGAs) is one of the effective ways to find botnets. Study on the vulnerability of deep learning models to adversarial attacks can enhance the robustness of DGA detection mechanism. This paper proposes CLETer, an improved DGA that provides a character-level evasion technique against state-of-the-art DGA classifiers. Based on existing DGA domain names, CLETer can intelligently generate adversarial examples by quantifying the influence of every character to the classification result and then changing the important characters. Those improved domain names can easily evade being detected and show good transferability. The experimental results demonstrate that when modifying only two characters, CLETer can effectively lower the LSTM classifier's recall from 99.76% to 1.29% and drop the CNN classifier's recall from 99.36% to 3.64%. It is proved that adversarial retraining is a viable defense strategy to CLETer.

Received on 14 Jan 2021; accepted on 09 February 2021; published on 18 February 2021

Keywords: cybersecurity, malware, domain generation algorithms, deep learning, adversarial attack

doi:10.4108/eai.18-2-2021.168723

1. Introduction

Malware has been evolved into the greatest threat to cybersecurity. As one of the most sophisticated modern malware, a botnet is a group of Internet-connected and malware-infected individual devices (bots) that receive operational instructions from command and control (C&C) servers directly controlled by attackers (botmasters). The botnet is used to conduct harmful cyber-attacks such as distributed denial-of-service [1], spam [2], information theft [3], etc. The implementation of these attacks depends on the information exchange between botmaster and bots. This is not only the core of botnet construction but also the key for the attack-defense game.

To evade detection and make botnets more robust, domain-flux is applied to establish a communication channel between bots and the C&C server. Domain-flux binds multiple domain names generated by domain generation algorithms (DGAs) to the IP address of the C&C server. Botmaster and bots share the same DGA. DGAs can dynamically produce a large number of candidate domain names based on a given seed including the current date/time [4], trending topic, random number, dictionary, etc. The possible seeds work as shared keys letting botmaster and infected bots generate the same domain list. Only a few domain names are selected to be registered for subsequent attacks, making them harder to be detected than traditional hard-coded malicious domain names. The detection of DGAs is an important way to prevent botnet attacks.

In the field of DGA detection, deep learning has been widely applied and achieved productive results. At the same time, its potential security problems have gradually attracted researchers' attention. From the perspective of botnet operators, it is beneficial to generate DGA domain names that can evade security detection. So that it's of great significance to research on generating adversarial examples against deep learning models for DGA detection. Recent papers propose some novel approaches. DeepDGA [5] makes use of Generative Adversarial Networks (GANs) to generate pseudo-random domain names that are more difficult

*Corresponding author. Email: yfang@scu.edu.cn

to be distinguished from benign ones. Charbot [6] generates adversarial domain names completely based on black-box and only needs to replace two random characters of existing benign domain names. While both the two ways can decrease the accuracy of DGA classifiers, they are not associated with the generated adversarial examples with the targeted model. Deception_DGA [31] utilizes the knowledge of manually engineered features to construct a new DGA, which can make the random forest classifier powerless but works weakly on a deep learning classifier. MaskDGA [8] is a black-box attack method that generates adversarial examples by training a substitutive model and based on the Jacobian-based saliency map. But the substitutive model must include a one-hot encoding layer to meet the requirements of the following calculation. Taken together, our primary goal is to carry out the adversarial attack in a black-box manner, explore the targeted model's vulnerability not depending on the knowledge of the model, and generate adversarial domain names to evade the deep learning DGA classifiers.

In this paper, we introduce CLETer, a character-level evasion technique to "fool" deep learning-based DGA classification models. It applies to the black-box attack scenario, where the attacker doesn't know the details of the targeted model (include network structures, parameters or training data, etc.), but can query the targeted model and get the output feedback of classification probability and label.

CLETer can intelligently generate adversarial domain names by modifying existing DGAs, including two steps: (1) evaluate every character's influence to the classification result of targeted model and (2) perform character-level transformers for those important characters. If a DGA domain name is detected as malicious initially but turned to benign finally, it's considered to construct adversarial example successfully.

To evaluate the effectiveness of CLETer, we choose two character-level deep learning models as our targets. We successfully apply CLETer to known DGAs and unknown DGAs. Those adversarial domain names can significantly reduce the recall of the two targeted models to below 10% when no more than three characters are transformed. To defense against CLETer's attack, we adopt the countermeasure of adversarial retraining that notably improves the robustness of targeted models.

Our main contributions are as follows:

- We propose a simple but effective evasion technique for deep learning DGA classifiers. It can perform character-level adversarial attacks in a black-box manner. And the adversarial examples generated by our method have good transferability in different deep learning models.
- We use scoring functions to quantify every character's influence on classification result, and verify its effectiveness to generate adversarial examples by modifying the important characters. We also verify that adversarial retraining is an effective defense strategy against our method.
- We reveal the vulnerability of state-of-the-art DGA classifiers to some simple adversarial attacks as CLETer and provide a noteworthy perspective to enhance the robustness of DGA detection mechanism.

2. Related Work

2.1. The Detection of DGAs

The detection of DGA domain names generally includes DNS traffic-based detection and textual-based detection. However, obtaining contextual information from network traffic is costly, thus it's difficult to be used for real-time monitoring and prevention. Adding randomness in the construction of DGA domain names makes them significantly different from legitimate ones. Therefore, detection based on the domain itself has become the mainstream method [9–12]. It can be divided into two types: one is the machine learning methods based on feature extraction [13], which mainly focus on the obvious differences in character distribution between legal and DGA domain names. These manually defined features include string length, entropy, vowel/consonant ratio, N-gram information, etc. However, feature extraction is a time-consuming task and these features are easily circumvented in detection, which results in high false positives and poor performance to detect new DGA domain names.

The other popular practice is the deep learning methods based on non-feature extraction. Compared with traditional machine learning, deep learning [14] shows huge superiority in automatic feature extraction from mass data. Remarkable achievements have been made in DGA detection [15–18]. The commonly used deep neural networks are LSTM networks and CNN. LSTM [19, 20] adds state information on the general RNN to enable it to learn long-term dependency information, and it is capable of capturing intercharacter sequential relationship. The CNN is a kind of deep neural network whose key structure is the convolution kernel. For the detection of DGA domain names, CNN can build a model at the character level and then extract the spatial structure features of DGA domain names with convolution operation.

2.2. The Adversarial Attack

Recent studies have shown that some deep learning models are vulnerable to adversarial attacks [21], that

is, subtle perturbations to inputs will lead detection models to get incorrect outputs. These artificially constructed instances by perturbing on originals are adversarial examples [22]. Adversarial attacks are attacks against the integrity of the Artificial Intelligence (AI) model, which are typically divided into two types: poisoning attacks [23, 24] against the data during the training phase, and evasion attacks [25] against the model during the test phase.

Evasion attack refers to the attack in which the attacker constructs specific input samples to fool the targeted system without changing the system. It can be classified in several ways. In different application scenarios, it is categorized into white-box attack [26] and black-box attack [27]. White-box attack requires obtaining the detailed information inside the machine learning model and then calculates directly to get the adversarial examples. Conducting the black-box attack depends on the transferability [27, 28] of adversarial examples. A substitutive classifier can be trained to perform the black-box attack [29]. From the perspective of attack results, evasion attacks can be divided into untargeted attack and targeted attack [30], the difference between them is whether to restrict the categories of adversarial examples. The untargeted attack only focuses on the successful attack results and doesn't limit the final category, but the targeted attack requires generated adversarial examples belonging to a specific category. In this paper, the detection of DGA domain names is essentially a binary categorization. Generated adversarial examples are misclassified as benign, which is considered as an untargeted attack.

In the study on adversarial examples of deep learning-based DGA detection methods. Deception_DGA designed by [31] is a novel DGA based on the white-box attack which can reduce the classification accuracy of the random forest classifier to 59.9% on a basis of requiring knowledge of manually engineered features used by the FANCI system [13]. But it works weakly on deep learning classifier.

MaskDGA [8] is a black-box attack method to evade DGA detection. It firstly trains a simple CNN substitutive model and then adds character-level perturbation to the input domain names based on the Jacobian-based saliency map. The resulting adversarial examples have achieved an obvious attack effect against four deep learning-based DGA classifiers.

With GANs being widely applied in cybersecurity, it is also gradually being used to the field of DGA classification. DeepDGA [5] is a generative model to generate pseudo-random domains that are more difficult to be distinguished from benign domain names and can effectively evade the detection of random forest DGA classifier using some manual-selected features. Extending these generated domain names to the training set can enhance the robustness of the classification model. Furthermore, Khaos [32] firstly make use of a Wasserstein Generative Adversarial Network (WGAN) to synthesize DGA domain names. It is easier to train than DeepDGA and performs more stably.

Compared with these new type DGAs, Charbot [6] is a much simpler approach of evasion attack against DGA detection. Charbot is completely based on black-box. It simply replaces two characters of the valid domain with characters having equal distribution in the DNS traffic, and generates effective DGA domain names to fool state-of-the-art DGA classifiers.

Coincident goals as ours, we try to evade detection by a simper but effective method. Thus we propose CLETer, an improved domain generation algorithm that achieves good evasion detection results only through simple character-level perturbation. CLETer can be implemented on any type of DGAs, liking putting an invisibility cloak on them to help them hard to be found.

3. Method

In a black-box attack scenario, targeted model's inner information is not attainable for us. When implementing CLETer, we only focus on the model's input-output relationship to explore its weakness. This is more applicable to real-world DGA detection.

Without changing the original network structure, CLETer can be applied as an extensional module deployed in the real-time detection of DGA (as presented in Figure 1). It requires the following main steps:

(i) prepare a list of DGA domain names that are seeking to evade detection.

(ii) determine the characters which have significant influence on classification result.

(iii) add character-level transform on them to evade a targeted deep learning model.

In this paper, we classify a domain name into two categories where benign domain names are labeled as "0" and malicious DGA domain names are labeled as "1". A domain name and its category are denoted in the form of $(\boldsymbol{x},y)$, where $\boldsymbol{x} = x_1x_2x_3...x_n$ is an input of character sequence and $y \in \{0, 1\}$. $x_i (i \in [0, n])$ represents the character on the i^{th} position and is from the following valid characters set C. In the character-level deep learning model, every valid character is a token as categorical features.

A deep learning-based DGA classification model is denoted as $F: \mathbb{X} \rightarrow \mathbb{Y}$, where F is a function mapping from the input domain name to its category. The final classification result is generally a probability value. In this paper, if the probability value is above 0.5, we

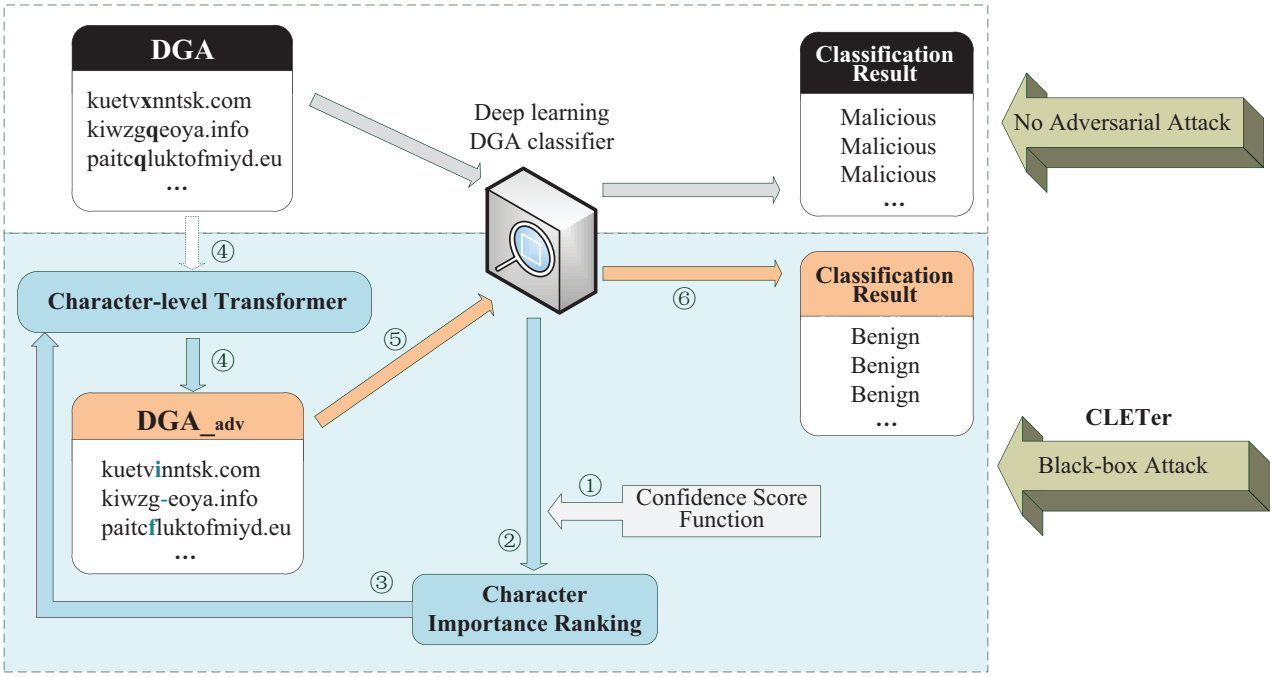

Figure 1. Implementation of CLETer. In the top part, DGA domain names are classified as malicious when no adversarial attack happens. In the bottom part, the adversarial examples generated by the black-box attack of CLETer are classified as benign. Firstly, for the existing DGA domain names, get the character importance ranking based on the confidence score function (①-②). Secondly, perform character-level transform to get adversarial domain names (③-④). Finally, the adversarial examples are misclassified as benign by targeted classifiers (⑤-⑥).

consider it is a malicious domain name, and if it's below 0.5, the domain name is labeled as benign.

$$C = \{'0','1','2','3','4','5','6','7','8','9','a','b','c','d','e', 'f','g','h','i','j','k','l','m','n','o','p','q','r','s','t', 'u','v','w','x','y','z','-'\}$$

3.1. Character Importance Ranking

In the deep learning DGA classification models, characters with different positions and content have different influence on the classification result. We use a score to quantify the importance of every character. The higher the score, the greater the influence on the classification results. And it is easier to mislead the DGA classifier by modifying those important characters.

For an input domain name $\boldsymbol{x} = x_1x_2x_3...x_n$, by removing the i^{th} character x_i, we can evaluate how the character x_i affects the classification results. Inspired by DeepWordBug [33] and TEXTFOOLER [34], we define four kinds of scoring functions:

(i) Head Influence Score (HIS);

$$HIS(x_i) = F(x_1, x_2, ..., x_{i-1}, x_i) - F(x_1, x_2, ..., x_{i-1})$$

The HIS quantifies a character's influence based on the preceding sequence. For the first character x_1, we define HIS (x_1) is 0.

(ii) Tail Influence Score (TIS);

$$TIS(x_i) = F(x_i, x_{i+1}, ..., x_n) - F(x_{i+1}, x_{i+2}, ..., x_n)$$

The TIS quantifies a character's influence based on the subsequent sequence. For the last character x_n, we define TIS (x_n) is 0.

(iii) Combined Influence Score (CIS);

$$CIS(x_i) = HIS(x_i) + \lambda TIS(x_i)$$

The CIS considers the combination of (1) and (2), where λ is a self-defined parameter and used to adjust the weight of HIS and TIS when calculating CIS.

(iv) Overall Influence Score (OIS).

$$OIS(x_i) = F(x_1, ..., x_i, ..., x_n) - F(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$$

The OIS evaluates a character's influence by comparing the overall difference before and after removing the character x_i.

We rank all the characters by descending order according to the output scores of targeted models.

3.2. Character-level Transformer

After calculating and ranking every character's score, the following step is to add character-level perturbation to construct adversarial examples. Here, we do this basically through substitution operation.

Our perturbation mechanism is based on the score ranking. Transforming the character with a higher score can easily confuse the targeted model to get the wrong result. We firstly substitute the character with a maximum score to another valid character in C and select the result with the lowest classification probability to substitute the character of the second-largest score. Continue transforming like that. We get a new domain name after every substitution and until the new domain name is classified as benign the process is stopped. The new one is the adversarial example we want to obtain. We can limit the maximum number of substitutions m. By applying different scoring functions mentioned in Section 3.1, the generation of adversarial examples through CLETer is summarized in Algorithm 1.

Algorithm 1 Adversarial Attack by CLETer

Require:

Input DGA domain name $\boldsymbol{x} = x_1x_2x_3...x_n$, category y, target model $F(\cdot)$, scoring function $S(\cdot)$, maximum number of substitutions m, valid characters set C.

Process:

$\boldsymbol{x_{adv}} \leftarrow \boldsymbol{x}$
for i = 1,2,...,n **do**
  scores(x_i) = S(x_i)
end for
Sort scores by the descending order to get a position index list $\boldsymbol{x_L}$: $x_{l_1}x_{l_2}x_{l_3}...x_{l_n}$
for j = 1,2,...,m **do**
  while $F(\boldsymbol{x_{adv}}) \rightarrow y$ is *malicious* **do**
    $\boldsymbol{x}^{'} \leftarrow$Transform $x_L[j]$ in $\boldsymbol{x_{adv}}$ to c in C
    if $F(\boldsymbol{x}^{'}) < F(\boldsymbol{x_{adv}})$ **then**
      $\boldsymbol{x_{adv}} \leftarrow \boldsymbol{x}^{'}$
    end if
  end while
end for
return $\boldsymbol{x_{adv}}$

The transform process needs to follow the rules of legitimate domain names. For example, "-" cannot be used in a beginning or end position.

We consider the adversarial examples generated successfully only if the classifier misclassifies the them as benign.

4. Evaluation

4.1. Experimental Setup

Datasets. To implement CLETer, we use open datasets from two sources:

- The Alexa one million domain names from *alexa.com*[1] are used as the benign (negative) dataset.
- The malicious (positive) dataset is from *360 DGA NetLab Open Project*[2]. It consists of over 40 DGA families, each data contains the domain name, the family it belongs and validation time.

This paper selects parts of these data (see Table 1). The 25 DGA families are divided into two parts: 18 DGA families (each family includes over 1000 domain names) are involved in training, used as *detected data* (*). The rest 7 DGA families not involved in training are used as *predicted data* (+).

In the classification of domain names, only the key part needs to be extracted. Therefore, the first step is data preprocessing. Generally, a complete domain consists of two or more parts that are separated by a dot (e.g. tsinghua.edu.cn). From right to left are top-level domain (TLD, e.g. cn), second-level domain (SLD, e.g. edu), third-level domain (e.g. tsinghua), etc. Few DGA families generate third-level domain, thus domain names in this paper are mainly referred to SLD part. But for some domain names with a third-level domain, we will pick the third-level domain part. Some examples are shown in Table 2.

Evaluation Metrics. The metrics used in experiments include AUC (Area Under Curve) and recall. AUC is an important index to evaluate the performance of a classifier. The recall is commonly used for the evaluation of DGA classification. It is defined as

$$Recall = \frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative}$$

Recall measures the model's ability to classify the correctly labeled instance of all DGA instances. Our goal is to mislead the classifier to predict adversarial examples as benign. The lower the recall, the better the evasion ability of CLETer.

[1]https://www.alexa.com/topsites/, Accessed: 2019-10-20
[2]https://data.netlab.360.com/dga/, Accessed: 2020-07-09

Table 1. Datasets for evaluating CLETer

Type	Total	LSTM		CNN	
		Train	Test/Attack	Train	Test/Attack
Benign					
Alexa	136,719	✓	T	✓	T
DGA Family					
gameover$_*$	10,000	✓	T/A	✓	T/A
locky$_*$	1,158	✓	T/A	✓	T/A
symmi$_*$	4,256	✓	T/A	✓	T/A
qadars$_*$	2,000	✓	T/A	✓	T/A
necurs$_*$	8,184	✓	T/A	✓	T/A
ranbyus$_*$	10,000	✓	T/A	✓	T/A
murofet$_*$	8,560	✓	T/A	✓	T/A
suppobox$_*$	2,252	✓	T/A	✓	T/A
virut$_*$	9,759	✓	T/A	✓	T/A
emotet$_*$	10,000	✓	T/A	✓	T/A
banjori$_*$	10,000	✓	T/A	✓	T/A
tinba$_*$	10,000	✓	T/A	✓	T/A
ramnit$_*$	10,000	✓	T/A	✓	T/A
simda$_*$	10,000	✓	T/A	✓	T/A
rovnix$_*$	10,000	✓	T/A	✓	T/A
pykspa_v1$_*$	10,000	✓	T/A	✓	T/A
shifu$_*$	2,547	✓	T/A	✓	T/A
shiotob$_*$	8,003	✓	T/A	✓	T/A
Totol	136,719				
pykspa_v2_fake$_+$	799	-	A	-	A
cryptolocker$_+$	1,000	-	A	-	A
chinad$_+$	1,000	-	A	-	A
matsnu$_+$	911	-	A	-	A
dyre$_+$	1,000	-	A	-	A
dircrypt$_+$	764	-	A	-	A
vawtrak$_+$	839	-	A	-	A
Totol	6,313				

$*$ denotes *detected data*, $+$ denotes *predected data*
T denotes the data used in test set, *A* denotes the data used to generate adversarial examples

Table 2. Data preprocessing examples

Original		After data preprocessing
google.com	→	google
wikipedia.org	→	wikipedia
sina.com.cn	→	sina
yahoo.co.jp	→	yahoo

4.2. Targeted Deep Models

CLETer could be implemented in any character-level DGA classifier. Here, we perform experiments on two deep learning models: LSTM and CNN.

Character-level LSTM Model. In this part, we adopt the classical LSTM-based DGA classifier [15] which firstly used LSTM for real-time detection of DGA domain names. Firstly, construct an encoding dictionary for characters in *C*. Every valid character corresponds to an assigned value (from 0 to 37). Secondly, the embedding layer transforms input domain names with different lengths to fixed-length feature vectors $\mathbb{R}^{d\times l}$, l is the maximum length determined by the training set, and d is a tunable parameter representing an

embedding. In our model, we choose $d = 128$. Then, the LSTM layer serves essentially as a feature extractor. In DGA binary classification, *sigmoid* is applied in the logistic regression layer. Besides, a dropout layer is used between the LSTM layer and logistic regression to prevent overfitting when training.

The *detected data* are involved in the training and test phase (see Table 1). Classification models are run using 10-fold cross-validation with a maximum of 25 epochs. Training set accounts for 80% and the other 20% are used as the test set. Validation accounts for 5% of the training set. Finally, We get the trained LSTM classifier with AUC is 0.996.

Character-level CNN Model. The other is a CNN-based DGA classifier. The embedding layer encodes the input domain name into a two-dimensional tensor. We perform multiple convolution operations *Conv(t, k, n)* to extract local features, and then this information is aggregated into a fixed-length feature vector. In this one-dimensional CNN, the size of filters $k = \{2, 3, 4, 5\}$, the number of convolution kernels $t = 256$, and n is input tensor. Max pooling is used to reduce the computational complexity of the model.

Training set and test set (see Table 1) account for 80% and 20% respectively. Finally, we get the trained CNN classifier with AUC is 0.995.

4.3. Adversarial Attack

We apply the targeted models in two cases: one is detecting the known 18 DGAs involved in training. We generate adversarial examples for the DGA domain names in the test set. The other is predicting the unknown 7 DGAs not involved in training. We generate adversarial domain names for them all.

Two scoring functions are adopted to calculate the character's score: *CIS (λ=1)* and *OIS*. For example, for a DGA domain name "quisleiymnnmilp", the calculation result of each character is showed in Figure 2.

Besides our method, we implement *Random Substitution (RS)* to generate adversarial examples: randomly select a character, and transform it into another random character. To enhance the reliability of the experiment, the random substitution process is repeated over 10 times. Average recalls are given.

4.4. Experimental Results on Classification

We evaluate the effectiveness of different adversarial methods on the two targeted models. The results of recalls are presented in Table 3.

Compared to the result when no adversary happens, we firstly find that targeted models have significantly lower recalls when classifying the adversarial examples generated by CLETer. The lower the recalls, the better the effect of evading DGA detection. Moreover, the method of *RS* has little effect on the final recalls compared to CLETer. This also proves that randomly modifying original DGA samples does not work and a well-founded selection of perturbation tokens when generating adversarial examples is feasible.

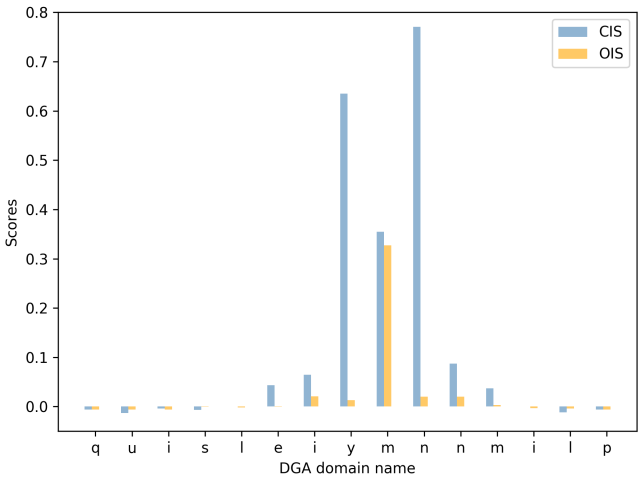

Figure 2. An example of calculating confidence scores.

CLETer is effective for both known and unknown DGAs. Choose different scoring functions to calculate importance scores has a different impact on classification results. As the number of substitutions m increases, the recalls decrease more significantly.

When applying CLETer on the 18 DGA families involved in training phase, the scoring functions of *CIS* and *OIS* are reliable to decide what important characters to be substituted. Here, *CIS* is superior to *OIS*. When two characters are transformed, the recalls reduce from 99.76% to 1.29% in the LSTM model and 3.64% in the CNN model.

When the two targeted models are predicting DGA families not involved in training model, the prediction results are slightly worse than detection results when no adversary happens. Because the deep learning networks did not learn the features of those DGAs. After implementing CLETer, the recalls also decrease significantly. *OIS* stands out a little. By twice substitutions, almost all the adversarial examples can evade the detection of the CNN classifier. Moreover, the method of *RS* hardly works in the adversarial attack.

Knowing how to select important characters to modify is the key to CLETer's phenomenal success. For *CIS* and *OIS*, the higher the score of a character is, the more significant effect it has on classification results. That is to say, changing these important characters is easy to cause misclassification.

To sum up, CLETer is effective in fooling deep learning classifiers that are used to detect known DGAs and predict unknown DGAs.

Table 3. The results of recalls for two targeted models

Targeted model	Data	Attack	The number of perturbed characters: *m*				
			1	2	3	4	5
LSTM (AUC = 0.996)	detected data$_*$	No adversary			**99.76%**		
		RS	95.74%	92.45%	88.84%	86.57%	84.86%
		CLETer_CIS	27.83%	1.29%	0.74%	0.37%	0.37%
		CLETer_OIS	51.22%	22.40%	11.39%	7.62%	5.91%
	predected data$_+$	No adversary			**81.58%**		
		RS	80.55%	80.40%	80.22%	80.04%	79.98%
		CLETer_CIS	43.01%	20.69%	14.07%	10.44%	8.02%
		CLETer_OIS	46.76%	21.70%	9.66%	4.65%	2.78%
CNN (AUC = 0.995)	detected data$_*$	No adversary			**99.36%**		
		RS	92.22%	89.34%	85.52%	82.78%	80.15%
		CLETer_CIS	3.69%	3.64%	3.64%	3.64%	3.64%
		CLETer_OIS	27.12%	3.67%	3.63%	3.63%	3.63%
	predected data$_+$	No adversary			**82.89%**		
		RS	80.71%	78.87%	77.39%	76.31%	74.70%
		CLETer_CIS	48.07%	23.01%	10.97%	5.96%	4.09%
		CLETer_OIS	17.26%	0.01%	0.00%	0.00%	0.00%

4.5. Transferability of Generated Adversarial Examples

To further verify the transferability of adversarial examples generated by CLETer, we use adversarial domain names generated through the LSTM model to attack the CNN model. These adversarial examples are obtained by modifying the 18 known DGAs in the test set of the LSTM model through the scoring functions of *CIS* and *OIS*. They have been tested by an LSTM classifier and can successfully evade its detection. Compared to other adversarial examples, they are more targeted and offensive.

The attack results are showed in Table 4. It's obvious to find that those adversarial examples cause extremely low recalls to the targeted CNN model. Over 96% of adversarial examples generated by changing only a character are misclassified. It means that using the data that has successfully attacked the known model to attack the unknown model can greatly improve the effectiveness. The adversarial examples generated through CLETer show good transferability in different deep learning models.

Table 4. The adversarial attack results on CNN model

Attack	Perturbed characters: *m*				
	1	2	3	4	5
No adversary			**99.36%**		
CLETer_CIS	3.88%	2.56%	2.33%	2.26%	2.22%
CLETer_OIS	3.66%	2.47%	2.22%	2.16%	2.14%

5. Evaluation of Adversarial Defense

Adversarial examples generated by CLETer have achieved noteworthy attacking results. To defense against the proposed evasion technique, we consider using *Adversarial Retraining* as the countermeasure. *Adversarial Retraining* is a proactive defense strategy[35], which can make neural networks more robust. In this experiment, we randomly select 5,000 samples from training set of the two targeted models and generate adversarial examples for them respectively. *CIS* and *OIS* scoring functions are used and the number of perturbed characters $m = 1$. Then, we augment the 5,000 adversarial domain names into the training set and retrain the classification model for another 2 epochs. Finally, we evaluate the retrained targeted models on a test set including 10,000 adversarial examples generated by CLETer ($m \leq 5$). The result of the recalls is presented in Table 5.

From the experimental results, the retrained LSTM and CNN models can effectively detect adversarial examples generated by CLETer. This also proves that the effectiveness of CLETer is not accidental but follows certain inherent rules and features, which has wide adaptability. Augmenting the training set with those

Table 5. The results of recalls for two retrained models

Targeted model	Defense	Attack		
		No adversary	CLETer_CIS	CLETer_OIS
LSTM	No defense	99.76%	6.60%	3.74%
	LSTM_retrain	99.58%	**91.12%**(↑84.52%)	**92.52%**(↑88.78%)
CNN	No defense	99.36%	2.26%	2.23%
	CNN_retrain	99.17%	**99.21%**(↑96.95%)	**99.02%** (↑96.79%)

adversarial examples and retraining the deep learning models indeed improve their defensiveness.

6. Discussion

It is sufficient to prove the vulnerability of the deep neural network that the well-designed classifiers can be destroyed only by subtle perturbation. In this paper, it is feasible to take the confidence score of the model as the basis of perturbation. But CLETer's successful application has been verified only on the character-level deep neural models. It is unknown whether it is valid to feature-based classifiers. Further research is needed.

CLETer is a character-level evasion technique by replacing characters on existing DGA domain names. The experimental results show the effectiveness of CLETer to generate adversarial examples, but our character-level perturbations could be improved through combining a more sophisticated algorithm such as semantic parsing, syntactic parsing, etc. Character-level operations also can be extended to swap, insertion, or deletion. Furthermore, considering more effective countermeasures to defend against attacks such as CLETer is a promising future work.

7. Conclusion

This paper proposed a simple but effective evasion technique for DGA domain names, we called CLETer. CLETer can intelligently generate adversarial examples for existing DGA domain names, including two steps: firstly, use the confidence score to quantify the influence of every character in a domain name to classification result. Secondly, transform the important characters to get a new domain name. We proposed two scoring functions to calculate the confidence score and proved their practicality in measuring the influence on the classification. The adversarial examples generated by CLETer effectively evade the detection of LSTM and CNN classifiers and reduce the recalls under 10%. CLETer is applied to a black-box manner where we don't know the structure and parameters of the targeted model but directly observe the relationship between input and output of it. Those adversarial domain names generated through CLETer show good transferability in different deep learning classifiers. We also proved the effectiveness of adversarial retraining to defense against the attack of CLETer.

The successful application of CLETer reveals the vulnerability of deep learning DGA classifiers to such so simple adversarial attacks. CLETer can be applied as an extensional module and be deployed flexibly. Generating adversarial examples by CLETer provides a noteworthy perspective to enhance the robustness of the DGA detection mechanism.

Acknowledgement. This work was supported by the National Key Research and Development Program (No.2016YFE0206700, No.2018YFB0804503) and the Fundamental Research Funds for the Central Universities.

References

[1] E. Alomari, S. Manickam, B. B. Gupta, Karuppayah, and R. S. Alfaris, "Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art," International Journal of Computer Applications, 49(7), 24–32, 2012.

[2] A. Pathak, F. Qian, Y. C. Hu, Z. M. Mao, and S. Ranjan, "Botnet spam campaigns can be long lasting: evidence, implications, and analysis," Performance evaluation review, 37, 13–24, 2009.

[3] A. Nadler, A. Aminov, and A. Shabtai, "Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol," Computers & Security, 2017.

[4] P. Daniel, Y. Khaled, K.Michael, B. Johannes, G. P. Elmar, "A comprehensive measurement study of domain generating malware," In Proceedings of the 25th USENIX Conference on Security Symposium. USENIX Association, USA, 2016, pp. 263–278.

[5] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-Tuned Domain Generation and Detection," In Proceedings of the ACM Workshop on Artificial Intelligence and Security. Association for Computing Machinery, New York, NY, USA, 2016, pp. 13–21.

[6] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, Anderson, N. Cock, and D. Martine, "CharBot:

A Simple and Effective Method for Evading DGA Classifiers," IEEE Access, pp. 91759-91771, 2019.
[7] 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 Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. Association for Computing Machinery, New York, NY, USA, 2019, pp. 1916–1923.
[8] L. Sidi, A. Nadler, and A. Shabtai, "MaskDGA: A Black-box Evasion Technique Against DGA Classifiers and Adversarial Defenses," arXiv:1902.08909, 2019.
[9] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting Algorithmically Generated Malicious Domain Names," In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10), Association for Computing Machinery, New York, NY, USA, pp. 48–61, 2010.
[10] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix DGA-based botnet tracking and intelligence," International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment Springer, Cham, vol. 8850, pp. 192–211, 2014.
[11] W. W. Zhang, J. Gong, Q. Liu, S. D. Liu, and X. Y. Hu, "Lightweight Domain Name Detection Algorithm Based on Morpheme Features," Journal of Software, vol. 27, pp. 2348–2364, 2016.
[12] D. Truong and G. Cheng, "Detecting domain-flux botnet based on DNS traffic features in managed network," Security and Communication Networks, vol. 9, pp. 2338–2347, 2016.
[13] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI: feature-based automated NXDomain classification and intelligence," In Proceedings of the 27th USENIX Conference on Security Symposium. USENIX Association, USA, 2018, pp. 1165–1181.
[14] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, The MIT Press: Cambridge, MA, USA, 2016.
[15] J. Woodbridge, H.S. Anderson, A. Ahuja, and D.Grant, Predicting domain generation algorithms with long short-term memory networks. arXiv:1611.00791, 2016.
[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, 2017, 275, 2401–2413.
[17] B. Yu, D. L. Gray, J. Pan, M. D. Cock, and A. C. A. Nascimento, "Inline DGA Detection with Deep Networks," In Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, 2017, pp. 683–692
[18] J. Huang, P. Wang, T. Zang, Q. Qiang, Y. Wang, and M. Yu, "Detecting Domain Generation Algorithms with Convolutional Neural Language Models," In Proceedings of the 2018 17th IEEE International Conference on IEEE International Conference on Trust, Security And Privacy In Computing And Communications, 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, 2018, pp. 1360-1367,
[19] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," Neural Computation, 2000, 12, 2451–2471.
[20] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks," Computational Intelligence, vol. 385, 2012.
[21] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, pp. 372-387, 2016.
[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv:1412.6572, 2015.
[23] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks," In Proceedings of the 10th international conference on Multiple classifier systems (MCS'11), Berlin, Heidelberg, pp. 350–359, 2011.
[24] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks against Support Vector Machines. In Proceedings of the 2012 29th International Coference on International Conference on Machine Learning (ICML'12). Omnipress, Madison, WI, USA, pp. 1467–1474, 2012.
[25] V. Yevgeniy and K.Murat, Adversarial Machine Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, 2018, 12(3), 1–169.
[26] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for NLP," In the Annual Meeting of the Association for Computational Linguistics, pp. 31–36, 2018.
[27] N. Papernot, P. Mcdaniel, I. J. Goodfellow, S. Jha, Z. B.Celik, and A. Swami, "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples," arXiv: 1602.02697, 2016.
[28] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Ferguset, "Intriguing properties of neural networks," arXiv: 1312.6199, 2013.
[29] N. Papernot, P. Mcdaniel, X. Wu, S. Jha, A. Swami, "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks," In 2016 IEEE Symposium on Security and Privacy (SP "16), 2016, pp. 582–597.
[30] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. L. Li, "Generating Textual Adversarial Examples for Deep Learning Models: A Survey," arXiv: 1901.06796, 2019.
[31] 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 Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), New York, NY, USA, 2019; pp. 1916–1923.
[32] X. Yun, J. Huang, Y. Wang, T. Zang, Y. Zhou, and Y. Z. Zhang, "Khaos: An Adversarial Neural Network DGA With High Anti-Detection Ability," IEEE transactions on information forensics and security, 2020, 15, 2225-2240.
[33] J. Gao, J. Lanchantin, M. L. Soffa, and Y. J. Qi, "Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers," IEEE Security and Privacy Workshops (SPW), 2018, 50–56.
[34] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is BERT Really Robust? Natural Language Attack on

Text Classification and Entailment," arXiv: 1907.11923, 2019.

[35] 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, pp. 2805–2824.