

MaskDGA: A Black-box Evasion Technique Against DGA Classifiers and Adversarial Defenses

Lior Sidi

Ben-Gurion University of the Negev
liorsid@post.bgu.ac.il

Asaf Nadler

Ben-Gurion University of the Negev
asafnadl@post.bgu.ac.il

Asaf Shabtai

Ben-Gurion University of the Negev
shabtaia@bgu.ac.il

ABSTRACT

Domain generation algorithms (DGAs) are commonly used by botnets to generate domain names through which bots can establish a resilient communication channel with their command and control servers. Recent publications presented deep learning, character-level classifiers that are able to detect algorithmically generated domain (AGD) names with high accuracy, and correspondingly, significantly reduce the effectiveness of DGAs for botnet communication. In this paper we present MaskDGA, a practical adversarial learning technique that adds perturbation to the character-level representation of algorithmically generated domain names in order to evade DGA classifiers, without the attacker having any knowledge about the DGA classifier’s architecture and parameters. MaskDGA was evaluated using the DMD-2018 dataset of AGD names and four recently published DGA classifiers, in which the average F1-score of the classifiers degrades from 0.977 to 0.495 when applying the evasion technique. An additional evaluation was conducted using the same classifiers but with adversarial defenses implemented: adversarial re-training and distillation. The results of this evaluation show that MaskDGA can be used for improving the robustness of the character-level DGA classifiers against adversarial attacks, but that ideally DGA classifiers should incorporate additional features alongside character-level features that are demonstrated in this study to be vulnerable to adversarial attacks.

CCS CONCEPTS

• Security and privacy → Intrusion/anomaly detection and malware mitigation.

KEYWORDS

Adversarial learning, Deep learning, Botnets, DGA

1 INTRODUCTION

Botnets are groups of inter-connected devices that are designed to carry large-scale cyber-attacks [51] such as distributed denial-of-service [2], data-theft [28], and spam [34]. The design of modern botnets often involve advanced techniques that challenge the ability to detect the botnet operation and take it down. Domain generation algorithms (DGAs) [36], a notorious technique, has been used by more than 40 documented botnets in the last decade.

Domain generation algorithms are used to generate a large number of pseudo-random domain names, which are usually based on the date and a secret input (seed). A bot and its command and control server that wish to communicate will both execute the DGA with a shared seed in order to generate a sequence of domain names and identify the one through which their communication can take place. DGAs can be used to generate thousands of domain names

per day which must be identified and analyzed in order to shutdown of the botnet.

The detection of algorithmically generated domain (AGD) names initially focused on capturing binary samples of bots, extracting their algorithms and seeds, and generating the domain names in advance for mitigation [35, 37, 43]. However, by using new input seeds, this approach can be easily evaded by botnets. In fact, between the years of 2017 and 2018 at least 150 new seeds were introduced by botnets,¹ a figure which is more than two times the number of documented DGAs. Using machine learning in order to inspect the lexicographic patterns of domain names for detecting AGD names and classify their generating algorithms [46, 52] is an alternative and more generalized approach.

Machine learning-based DGA detection techniques have become extremely successful, with state of the art algorithms achieving high detection rates on multiple datasets [42] with inline latency [23]. These techniques reduce the effectiveness of DGAs for maintaining resilient and stealthy botnet communication. From the botnet operator’s perspective, a solution that can evade these state of the art detection mechanisms by using adversarial machine learning to produce AGD names that are less likely to be detected can be beneficial.

Adversarial machine learning is a technique in the field of machine learning which attempts to “fool” models (during either the training or execution phases) through adversarial input [25], also referred to as adversarial samples.

In the context of DGA classification, generative adversarial networks (GANs) [18] were previously suggested by Anderson *et al.* [3] in a method called DeepDGA, in order to generate adversarial samples of “fake” domain names that are indistinguishable from real benign domain names that were available in the training set. The fake domain names generated are used to augment the training set with additional benign domain names and accordingly improve the model’s robustness. Because DeepDGA was trained to generate domain names that resemble benign domains, the fake domain names generated are mostly short and readable. This makes the generated domain names impractical for botnets, since they are more likely to be used by legitimate users (e.g., the domain name “laner.com” generated by DeepDGA was already owned by Laner Electric Supply). In addition, short domain names are usually more expensive and thus increase the attacker’s costs for maintaining the botnet communication.

In this paper we explore the ability to evade detection by DGA classifiers. We present MaskDGA, a black-box adversarial learning technique that adds perturbations to a character level representation of an AGD and transforms it to a new domain name that is falsely classified as benign by DGA classifiers, without prior knowledge

¹Based on the change-log of <https://data.netlab.360.com/dga/>

of the classifier’s architecture and parameters. MaskDGA can be applied as an extension to an existing malware to evade detection on the network perimeter by security solutions (as depicted in Figure 1).

MaskDGA utilizes the transferability property [30] and is conducted by first training a substitute model on datasets of publicly known AGD names. The attacker then generates a set of AGD names to which MaskDGA is applied. The domain names generated are provided as input to the substitute model and applied in a single feed forward step. The results of the feed forward step are used to compute the loss with regards to the benign class. MaskDGA then performs a single back-propagation step in which the loss propagates the gradients back to the input to form a Jacobian saliency map [32] (JSM). Finally, for every domain name generated, MaskDGA replaces exactly half of the characters that had the largest gradient values in the JSM, so that the resulting domain name remains long and likely unreadable. Changing only half of the characters for a set of randomly generated AGD names also implies that we can expect a small number of collisions among the domain names in the output set, and therefore, the number of domains in the output set will remain close to that of the original set (created by the DGA). Eventually, the effort required for detecting and taking down the botnet remains the same.

MaskDGA was designed based on the Jacobian-based saliency map adversarial attack [32] (JSMA) that was originally applied to images. In contrast to the JSMA attack, MaskDGA is restricted to (1) producing valid character-level changes, (2) performing a constant number of changes, and (3) changing the same feature (character) only once, thus making it more suitable for character-level DGA classification evasion.

The evaluation of MaskDGA was performed using the recently published DMD-2018 dataset of AGD names and open-source DGA classifiers [12]. We also applied DeepDGA [3] as a technique for evading the detection models, and compared its performance with that of MaskDGA. The results show that when applying MaskDGA, the detection rate degrades from an average F1-score of 0.977 to 0.495, while the DeepDGA attack results in an average F1-score of 0.780.

We also evaluated MaskDGA against the same DGA classifiers enhanced with adversarial learning defenses, namely adversarial re-training (using domains generated by DeepDGA [4] and by MaskDGA) and distillation [33], and assessed their effectiveness. The results shows that adversarial re-training can be effective to some extent against the evasion techniques but still leads to the conclusion that a more resilient detection mechanism is required.

In summary, the contributions of this paper are as follows:

- we present a practical evasion technique for DGA detection machine learning models;
- the evasion technique can be applied as a wrapper on the existing DGA source code used by the botnet and therefore easy to apply;
- the technique is generic and can be applied to any DGA family and input seed, while adhering to the expected requirements of a DGA, i.e., uniqueness, validity and rareness of domain names;

- we present an evaluation of the presented evasion technique using four recently published DGA classifiers, and we assess the robustness of MaskDGA against commonly used adversarial defense techniques;
- from the presented evasion technique we conclude that the detection of DGA mechanisms should not rely solely on character-level DGA features and ideally DGA classifiers should incorporate additional behavioral features.

2 BACKGROUND

2.1 Character-level DGA classification

A character-level representation of a word W over the alphabet of symbols V is a binary matrix $X_{|W| \cdot |V|}$ s.t.

$$\forall i \in [W], j \in [V] : X_{i,j} = \begin{cases} 1 & W_i = V_j \\ 0 & \text{otherwise} \end{cases}$$

i.e., $X_{i,j}$ is set to one if the i -th character of the word W matches the j -th symbol in the alphabet V and zero if otherwise.

Recently published models for DGA classification [40, 52] are based on character-level representations of domain names due to their high accuracy [6] and inline latency [23], provided that no additional context is required (e.g., network traffic features). Therefore, the focus of the evasion techniques presented in this study is on character-level DGA classification.

2.2 Adversarial learning

Adversarial learning attacks can be divided into poisoning attacks performed during training [7, 9], and evasion attacks performed during testing [8, 19, 31]. Evasion attacks usually involve adding perturbation (noise) to the input sample’s features to result in a misclassification by the attacked model. Such inputs are also referred to as adversarial samples. In order to make the targeted model classify the sample X whose true label is Y as Y^* , the attacker would have to find a perturbation δ_X such that $(X + \delta_X)$ is classified as Y^* . If the attack is aimed at a specific class in the data, it is referred to as a targeted attack [30].

Attacks also vary based on the knowledge the adversary has about the classifier. A black-box evasion attack requires no knowledge about the model beyond the ability to query it as a black-box, i.e., inserting an input and getting the output classification. In a white-box attack, the adversary can have varying degrees of knowledge about the model architecture, hyper parameters and data used for training.

Most black-box attacks rely on the concept of adversarial example transferability, also known as the transferability property [30]. According to the transferability property, adversarial examples crafted against one model are also likely to be effective against other models, even when the models are trained on different datasets (but assumed to be sampled from the same distribution) [32, 45]. This means that the adversary can train a substitute model, which has decision boundaries similar to the original model, and perform a white-box attack on it, i.e., obtain the back-propagated gradients in order to add perturbation that would result in the desired target misclassification. Adversarial examples that successfully fool the substitute model are likely to fool the original model as well [31].

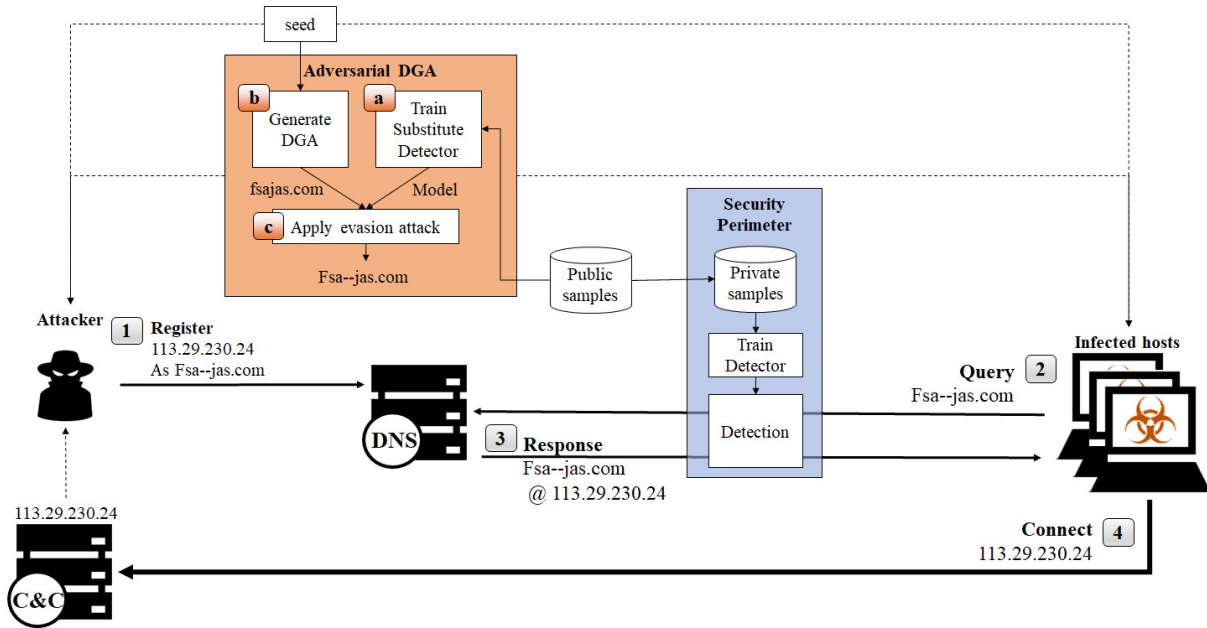


Figure 1: After training a substitute model (a), and generating a set of AGD names using an existing DGA (b), the attacker applies MaskDGA as a wrapper to add perturbation to the set of AGD names, which will result in a new set of adversarial domain names that evade detection (c). The bot operator can register the adversarial AGD names (1) to allow its bots to establish a communication channel with their command and control server via the adversarial domain names (2-4) while evading detection on the security perimeter.

In this paper we present a black-box technique aiming at evading DGA detection models.

3 THREAT MODEL

Training capability (black-box). We assume that the targeted DGA detection models are not available to the botnet operator, and therefore the attacker cannot rely on knowledge about the architecture of the targeted model. However, we assume that the botnet operator can use publicly available AGD datasets and previously published architectures in order to train a substitute model.

Targeted misclassification for the benign class. The targeted models are multiclass DGA classification models whose input is a domain name and output is a probability distribution for the set of DGA classes and the benign class (e.g., [40, 52]). The class with the maximal value in the output probability distribution is selected by the DGA classifier as the predicted class. A misclassification between the various DGA labels would still result in the detection of the malicious domain name and is therefore considered an unsuccessful adversarial attack. Thus, the attacker’s goal is a targeted misclassification of the adversarial input as the benign class.

4 THE EVASION TECHNIQUE

4.1 Overview

The evasion technique is performed according to the following main steps (portrayed in Figure 2):

- **Training a substitute model.** The attacker acquires a dataset of public samples of AGD names and uses it to train a discriminative model that distinguishes AGD names from benign domain name. The model is referred to as the substitute model.
- **Generating the algorithmically generated domain names.** The attacker executes an existing DGA with a randomly selected seed and generates a set of AGD names.
- **Constructing a Jacobian-based saliency map.** For every generated AGD, the attacker performs a single feed forward step in the substitute model. The results of the feed forward step are used to compute the loss with regard to the benign class. The attacker performs a single back-propagation step on the loss in order to acquire the Jacobian-based saliency map, which is a matrix that assigns every feature in the input AGD with a gradient value. Features (characters) with higher gradient values in the JSM would have a more significant (salient) effect on the misclassification of the input AGD name and hence the name saliency map.
- **Adding perturbation to the input AGD name based on the JSM.** For every input AGD name, iterate over every character position from the first character to the last. For every position, save the highest salient symbol and its value, e.g., in the first position the symbol “a” may have the highest value in the JSM. Set the top half of the character positions with regard to the maximal saliency values to the symbols that yielded the maximal saliency values (as portrayed in Algorithm 1).

4.2 Training a substitute model

The substitute model is a discriminative model (i.e., classifier) that distinguishes algorithmically generated domain names from benign domain names. The input layer of the model accepts character-level domain name features in the feed forward step and then reads the gradient for every feature in the back-propagation step for the construction of the JSM. The output layer of the model can be either binary (e.g., DGA or benign) or multiclass to predict the specific DGA. Formally, the substitute model learns a function F that accepts an input domain name X with a predicted class Y . These notations are later used in Algorithm 1.

4.3 Generating the domain names

The generation of original AGD names can be executed either by botnets that already have a DGA mechanism implementation and are looking to extend it to evade detection, or by a new DGA implementation. Although the evasion technique works regardless of the selected DGA, the attacker should take into account two considerations: the average length and the number of generated domain names.

The average length of the generated domain names represents a trade-off between the ease of evasion and the availability of the domain name generated for botnet communication. AGD names that are shorter (usually less than eight symbols) have a higher chance of being readable. Readable domain names have greater likelihood of evading detection, but on the other hand, they are more likely to be owned by legitimate users and thus unavailable for attacks. In addition, MaskDGA might make the provided domain names even more readable (while maintaining their length) and accordingly less available. In contrast, longer domain names are likely to be available and inexpensive. However, DGA classifiers are sensitive to the length of the domain names and are more likely to predict longer than average domain names as AGD names, regardless of their characters.

The number of generated domain names represents another trade-off between stealthy communication and resilient communication. DGAs that produce a very large number of domain names (e.g., thousands per day) are more difficult for take-down but are easier to identify because of the overall noise. An alternative is a DGA that produces a smaller number of domain names to choose a more stealthy communication. For instance: Banjori, Conficker and Gameover Zeus DGAs which generate thousands of domain names per day are ideal in terms of resilience, while Torpig and Matsnu generate up to three domain names per day and are better for stealthy communication [36].

4.4 Construct a Jacobian-based saliency map

The construction of the JSM begins by providing the substitute model with an AGD name X as performing a single feed forward step. For every AGD name, the feed forward step would provide a predicted probability distribution for the output classes of the substitute model. Based on the predicted probability distribution and the benign class distribution, we compute the *targeted* loss function $\mathcal{L}(F(X), Y^*)$ of the predicted value $F(X)$ with respect to the benign class Y^* . The back-propagation algorithm is then applied on the computed loss from the last hidden layer to the input layer

so that every feature of the input X is assigned with a gradient. For example, the feature $X_{i,j}$ will be assigned with the real-value gradient $\frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{0,0}}$.

Formally, the JSM (denoted as S) is the set of the back-propagated gradients for every input feature:

$$S = \begin{bmatrix} \frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{0,0}} & \frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{1,0}} & \cdots & \frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{|W|,0}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{0,|V|}} & \frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{1,|V|}} & \cdots & \frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{|W|,|V|}} \end{bmatrix}$$

4.5 Adding perturbation to the input AGD based on the JSM

The input character-level AGD X is added with a perturbation δ_X so that following constraints are met:

- (1) $X + \delta_X$ is a valid character-level representation of a domain name (see Subsection 2.1)
- (2) Exactly half of the characters of X are replaced

The selection the perturbation δ_X and the addition of its to transform X to X' is formally described in Algorithm 1. Initially, the algorithm computes the maximal value for every column in the JSM. The median of maximal value acts as a threshold and is henceforth denoted as Θ . MaskDGA replaces the character in position i to the j -th symbol in the alphabet if and only if the maximal value for column i of the JSM is j and $j > \Theta$. The character replacement is performed by adding a perturbation δ_X in which:

- (1) the original symbol is set to (-1) ,
- (2) the new symbol is set to $(+1)$, and
- (3) the rest of the symbols are set to zero.

By definition, since Θ is the median of the maximal columns in the JSM that correspond to character positions, exactly half of the characters are replaced. The result is a new domain name X^* which is the original X with the perturbation δ_X that was induced by the algorithm.

Algorithm 1 Adding perturbation to an algorithmically generated domain name based on the JSM

INPUT:

X - a character-level AGD name of size $|W| \cdot |V|$

Y^* - the targeted class

F - is the function learned by the substitution model

- 1: Let S be the saliency map of $\nabla F(X)$ w.r.t. Y^*
 - 2: $X^* \leftarrow X$
 - 3: **for** i in $0, \dots, |W|$ **do**
 - 4: $S_{j^*}^i \leftarrow \max_j S_{i,j}$
 - 5: Let Θ be the median of $\{S_{j^*}^0, \dots, S_{j^*}^{|W|}\}$ ▶ Set the threshold
 - 6: **for** i in $0, \dots, |W|$ **do**
 - 7: **if** $\operatorname{argmax}_j S_{i,j} \geq \Theta$ **then**
 - 8: $X_{i,j}^* = 1$
 - 9: $X_{i,\bar{j}}^* = 0$ for all $\bar{j} \neq j$
 - 10: **return** X^*
-

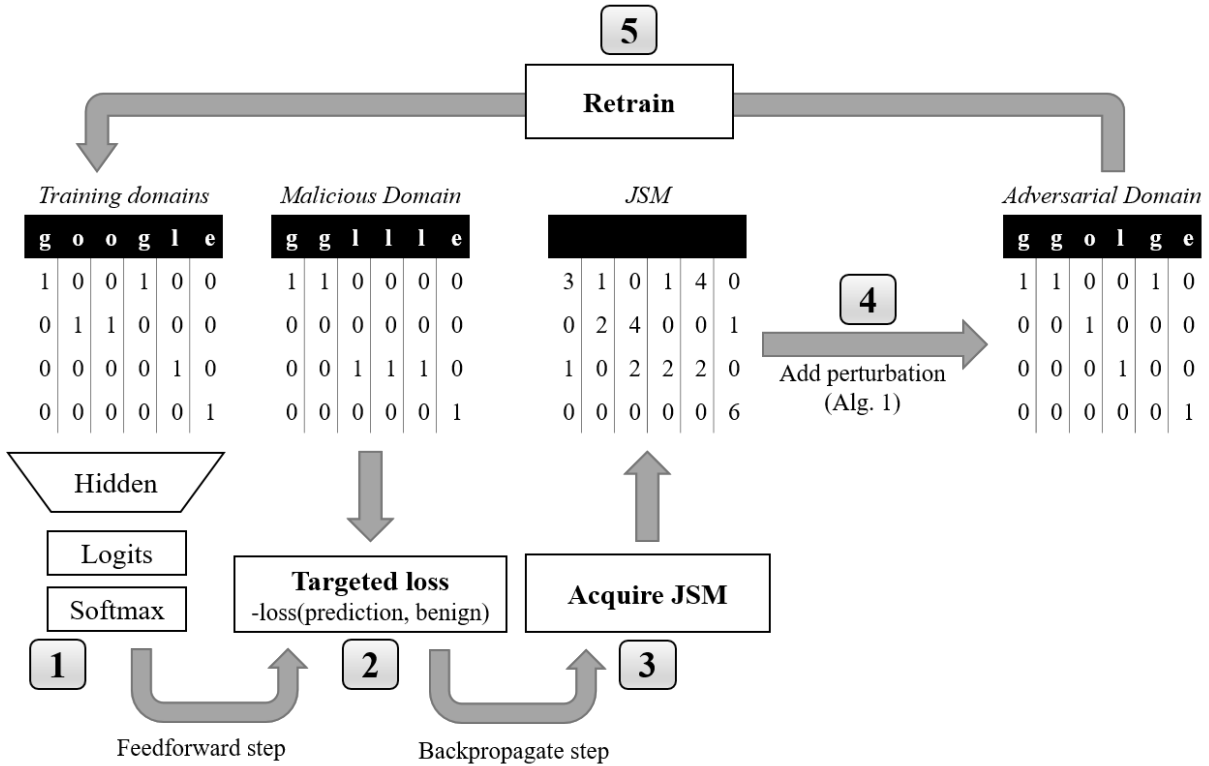


Figure 2: The attacker trains a substitute model on a publicly available dataset of AGD (1). A single feed forward step is performed using the substitute model on an input AGD name to compute the targeted loss with regard to the benign class (2). The targeted loss with regards to the benign class is used to back-propagate the gradients and construct the Jacobian-based saliency map (3) and add perturbation to the input AGD based on the JSM to result in a new adversarial domain names that evades detection (4).

5 EVALUATION

5.1 Overview

In this section we present the experiments conducted in order to evaluate the MaskDGA evasion technique. MaskDGA was tested on four recently presented models which are based on deep learning DGA classifiers using a public dataset of AGD names.

The experiments focused on evaluating the effectiveness of MaskDGA while targeting the attacked models in general, as well as when focusing on a specific DGA family.

The performance of MaskDGA was compared against two baseline attacks: (1) a random attack, which selects half of a domain’s characters at random and replaces them with uniformly selected characters from the alphabet of English letters, digits, hyphens and underscores, and (2) an attack based on DeepDGA [3].

The DeepDGA-based attack was conducted using the source code provided by the authors of [3] with several modifications that were required in order to use it for the attack. After upgrading the source code so we could use the newer version of the Keras library [11] and applying it to a recent Alexa top 500,000 sites (as was done by Anderson *et al.* [3]), the generator produced a limited variety of samples. This is a common challenge in applying GANs, also known as *mode collapse*, which expresses the sensitivity of GAN

architectures to minor changes in the configuration such as when using a different dataset [21]. Our understanding of the problem was that the generator was quickly over-fitted to the discriminator and failed to generate new samples. To overcome the mode collapse state we provided the discriminator with an initial advantage over the generator by pre-training the discriminator on a preliminary set of domain names that were generated by the generator before the training phase.

5.2 Datasets

The main dataset used for the evaluation is the DMD-2018 dataset [27, 47–49] which contains 100,000 benign domain names and 297,777 algorithmically generated domains that were evenly produced by twenty different DGA families (see Table 1). The substitute model that used in MaskDGA was trained on 113,335 randomly selected domain names (28.9% of the dataset). The targeted models (i.e., the four DGA classifiers) that we attempted to evade were trained on 264,456 domain names (67.6% of the dataset) to provide the DGA classifiers with an advantage against the attack. The remaining 9,986 domain names (3.3% of the dataset) were used to test the effectiveness of MaskDGA. Every domain name in the test set was perturbed and evaluated against the targeted models.

Additionally, the alexa.com dataset of top sites² was used to train the initial autoencoder of DeepDGA [4] as conducted by Anderson *et al.* [3]. Using the trained DeepDGA, we generated 500 adversarial domain names that were used to compare with MaskDGA, and an additional 63,500 domain names were used for the a defense as explained in Subsection 6.

Table 1: The dataset used for evaluating MaskDGA.

Class	Substitute training	Targeted training	Testing
Benign			
benign	27000	63000	-
DGA			
banjori	4349	10148	503
corebot	4349	10148	503
dircrypt	4349	10148	503
dnschanger	4349	10148	503
fobber	4349	10148	503
murofet	4349	10148	503
necurs	3704	8644	429
newgoz	4349	10148	503
padcrypt	4349	10148	503
proslikefan	4349	10148	503
pykspa	4349	10148	503
qadars	4349	10148	503
qakbot	4349	10148	503
ramdo	4349	10148	503
ranybus	4349	10148	503
simda	4349	10148	503
suppobox	4349	10148	503
symmi	4349	10148	503
tempedreve	4349	10148	503
tinba	4349	10148	503
Total DGA	86335	201456	9986
Total	113335	264456	9986

The number of domain names in the dataset that are used for the evaluation. The substitute training set is used by the attacker to train the substitute model. The targeted training set is used to train a DGA classifier model that is targeted by the attacker. The test set contains an additional set of adversarial AGD names that are tested against the DGA classifiers.

5.3 Substitute model architecture

The architecture of the substitute model has the character-level (one hot encoding) representation of the domain as the input layer and an output neuron for every DGA family or benign domain as the output layer. The hidden layers consist of one-dimensional CNN filters of size 2, 3, 4, 5, and a max pooling layer that reduces the domain length to one half of its original and two additional fully-connected layers. This architecture was chosen because we wanted to evaluate the effectiveness of MaskDGA when using a

model that is considerably weaker than the attacked DGA detection classifiers, and because it is lightweight and thus practical to apply.

The training of the substitute model was conducted for five epochs with batches of 128 domain names until convergence using the ADAM optimizer [22] with a learning rate of 0.01.

The implementation of the evaluated adversarial evasion technique was achieved using the Keras library [11] and the TensorFlow library [1] for the substitute model and the Cleverhans library [29] for computing the gradients based on the targeted loss, and accordingly the JSM.

The character-level representations inputs for domain names are based on the vocabulary of English letters, digits, hyphens, underscores and dots. We (hard-coded) restrict the MaskDGA from adding perturbations that would change a non-dot character to a dot character. This restriction is designed to avoid the *shortening* of the second-level domain (e.g., “example.com” turns into “exa.mple.com”) which has undesired effects on the effectiveness of the evasion technique (as explained in Section 1).

5.4 Targeted (attacked) models’ architecture

The targeted models that MaskDGA tries to evade from detection are based on the following recently proposed DGA classifiers:

- Endgame [3] which is based on a single LSTM layer (denoted as LSTM[Endgame])
- CMU [15] which is based on a forward LSTM layer and a backward LSTM layer (denoted as biLSTM[CMU])
- Invincea [40] which based on parallel CNN layers (denoted as CNN[Invincea])
- MIT [50] which based on stacked CNN layers and a single LSTM layer (denoted as CNN + LSTM[MIT])

The targeted models were implemented using the TensorFlow library [1] and are based on the source code that was published by [6] and trained on the the set of domain names that are reserved for the targeted model training in the dataset.

5.5 Evaluation metrics

The metrics used to evaluate the effectiveness of the evasion technique on the targeted models are precision, recall, and the F1-score, as is commonly used for DGA classification evaluation. Based on the threat model (see Section 3) where the goal is defined as targeted misclassification, we define the correct class as any DGA, instead of the specific DGA i.e., the classifier might predict the wrong DGA class but it is regarded correct as long as it did not misclassify an AGD as benign.

Precision computes the model’s ability to assign the correct label out of all the instances in cluster label.

$$Precision(C, Y) = \frac{\text{correct class}}{\text{predict class}} \quad (1)$$

Recall computes the model’s ability to retrieve all of the instances of a certain class; i.e., the model’s ability to retrieve all of the domains of a certain family.

$$Recall(C, Y) = \frac{\text{correct class}}{\text{entire class}} \quad (2)$$

²<https://www.alexa.com/topsites>

F1 score combines precision and recall together into one score using harmonic mean

$$F1(C, Y) = 2 * \frac{\text{precision}(C, Y) * \text{recall}(C, Y)}{\text{precision}(C, Y) + \text{recall}(C, Y)} \quad (3)$$

5.6 Results

Table 2 presents the precision, recall and F1 measures when applying the evasion techniques against the four tested models (LSTM[Endgame], biLSTM[CMU], CNN[Invincea], and CNN + LSTM[MIT]). We compare the performance of the targeted models when no attack is applied (*NoAttack*), when a random attack is performed (*Random*), when a DeepDGA-based attack is performed (*DeepDGA*), and when executing the presented evasion technique (*MaskDGA*).

Based on the results, we can see that when no adversarial attack is applied (*NoAttack*), all of the models perform almost perfectly on the test set. More specifically, the targeted models yield an average precision (0.977), recall (0.977) and F1-score (0.977), thus supporting the claim that recently published, deep learning based DGA classifiers are very accurate in detecting AGD names.

The *Random* attack shows a degradation in the average precision (-19%), recall (-26%), and F1-score (-26.5%), compared to the *NoAttack* case. This degradation in the evaluation measures indicates that the four targeted models are very sensitive to changes in the lexical patterns of DGA-based domain names.

The DeepDGA attack consists of generating pseudo-random domain names that are sampled from the same distribution of the benign Alexa top 1M domains using a GAN network. While clearly degrading the performance of the detection models in comparison to the *NoAttack* case, the DeepDGA generated domains are effectively detected by all four models with a higher average precision (+6%), recall (+5%) and F1-score (+0.3%) compared to the *Random* attack. The generation of domain names by DeepDGA relies on sampling symbols from a pre-trained multinomial regressor that provides the likelihood for every symbol at a given position. The sampling is performed independently for every position, and thus it is not guaranteed that the output domain name would not appear to be algorithmically generated. The MaskDGA evasion technique dominates all evaluated attacks on all four targeted models with an average recall rate of 0.575; i.e., an average DGA classifier would miss 42.5% of the AGD names compared to only 2.5% in the *NoAttack* case.

We were also interested in evaluating the effectiveness of the evasion technique on specific DGA families that are characterized by different patterns.

Table 3 presents the F1-score (which reflects the harmonic mean between the precision and recall) for the twenty DGA families that are available in the test set.

When *NoAttack* is applied, a similar average F1-score for all DGA families is observed. This indicates that the models memorize the lexicographical patterns of observed domains regardless of their length.

The *Random* attack, when focusing on the specific DGA families, results in a more significant degradation for some DGAs (e.g., Simda, Suppobox) than others (e.g., CoreBot, Symmi). The

Table 2: Evaluation results.

Targeted model	Precision	Recall	F1-score
CNN[Invincea]			
No Attack	1.00	1.00	1.00
Random	0.83	0.74	0.73
DeepDGA	0.91	0.80	0.84
MaskDGA	0.68	0.52	0.39
LSTM[Endgame]			
No Attack	0.98	0.98	0.98
Random	0.83	0.79	0.78
DeepDGA	0.74	0.73	0.74
MaskDGA	0.73	0.60	0.53
biLSTM[CMU]			
No Attack	0.98	0.98	0.98
Random	0.85	0.81	0.81
DeepDGA	0.75	0.82	0.78
MaskDGA	0.72	0.61	0.55
CNN + LSTM [MIT]			
No Attack	0.95	0.95	0.95
Random	0.80	0.77	0.77
DeepDGA	0.68	0.92	0.74
MaskDGA	0.63	0.57	0.51

The evaluation metrics of the targeted DGA classifiers on the test set when no attack is applied (*NoAttack*), a random attack is performed (*Random*), a DeepDGA-based attack is performed (*DeepDGA*), and when executing the presented evasion technique (*MaskDGA*).

Suppobox DGA is a dictionary-based DGA and therefore its generated domains appear as normal English terms (e.g., tablethirteen.net). Therefore, a random replacement of half of the characters would completely demolish the lexicographical pattern of Suppobox domains and accordingly the *Random* attack is highly effective for Suppobox. However, for DGAs such as CoreBot which produces long domains with uniformly selected characters (e.g., i8a0q2wdu8otulkfylo2gdq.ddns.net) the *Random* attack is not effective against all models (e.g., LSTM[Invincea] results with an average F1-score of 0.91).

The presented MaskDGA evasion technique is dominant over all DGA families across all targeted models. When MaskDGA is applied to the Suppobox dictionary-based DGA all targeted models detect the adversarial domain name with an average F1-score of 0.495. For the more difficult case of the corebot DGA, the average F1-score is 0.7725, thus arguing that while the longer AGD have more difficulty in evading detection, MaskDGA still manages to evade detection.

Note that in this analysis the DeepDGA could not be evaluated (and thus was omitted) because its input is a seed string and not AGD that were produced by DGA processes.

Finally, Table 4 contains examples of AGD that were generated by several DGA families and their adversarial form after adding perturbation by MaskDGA in order to provide further intuition. As

Table 3: Evaluation results per DGA family (average F1-score).

Model/DGA	banjori	corebot	dircrypt	dnshchanger	fobber	murofet	neurus	newgoz	padcrypt	prosilikefan	pykspa	qadars	qakbot	ramdo	ranbyus	simda	suppobox	symmi	tempedreve	tinba
CNN[Invincea]																				
No Attack	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
Random	0.78	0.91	0.74	0.79	0.79	0.89	0.78	0.88	0.72	0.81	0.80	0.77	0.83	0.78	0.73	0.7	0.70	0.92	0.77	0.76
MaskDGA	0.49	0.67	0.50	0.52	0.49	0.64	0.50	0.56	0.49	0.51	0.51	0.55	0.54	0.54	0.49	0.51	0.49	0.49	0.51	0.53
LSTM[Endgame]																				
No Attack	0.90	0.90	0.89	0.89	0.90	0.90	0.88	0.88	0.90	0.89	0.89	0.9	0.9	0.9	0.9	0.89	0.89	0.9	0.9	0.9
Random	0.78	0.89	0.69	0.63	0.84	0.84	0.76	0.85	0.71	0.75	0.73	0.78	0.81	0.84	0.83	0.65	0.69	0.88	0.75	0.74
MaskDGA	0.48	0.82	0.55	0.51	0.67	0.68	0.6	0.65	0.51	0.56	0.58	0.64	0.64	0.63	0.64	0.52	0.51	0.68	0.63	0.59
LSTM[CMU]																				
No Attack	0.87	0.87	0.87	0.86	0.87	0.87	0.85	0.87	0.87	0.86	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
Random	0.8	0.87	0.78	0.73	0.81	0.81	0.66	0.85	0.71	0.69	0.74	0.77	0.81	0.84	0.71	0.76	0.7	0.87	0.78	0.67
MaskDGA	0.49	0.86	0.61	0.58	0.64	0.66	0.53	0.67	0.49	0.53	0.57	0.58	0.63	0.6	0.55	0.57	0.5	0.81	0.62	0.54
CNN[MIT]																				
No Attack	0.76	0.76	0.75	0.76	0.76	0.76	0.74	0.75	0.76	0.75	0.75	0.76	0.76	0.76	0.76	0.75	0.76	0.76	0.76	0.76
Random	0.66	0.76	0.63	0.59	0.71	0.72	0.64	0.74	0.63	0.65	0.63	0.69	0.69	0.71	0.67	0.59	0.64	0.75	0.65	0.64
MaskDGA	0.48	0.74	0.5	0.49	0.56	0.62	0.53	0.63	0.48	0.54	0.51	0.52	0.56	0.51	0.53	0.49	0.48	0.64	0.52	0.51

The average F1-score of the targeted DGA classifiers on twenty DGA families when: no attack is applied (*NoAttack*), a random attack is performed (*Random*), and when executing the presented evasion technique (*MaskDGA*).

described in Section 4, the characters that are replaced by MaskDGA are the ones that have the highest saliency with regards to a misclassification as the benign class. Accordingly, the provided examples inflict two interesting trends. The clearest trend is that the underscore (“_”) characters is frequently used by MaskDGA since it is valid and common for benign domain names that appeared on the DMD-2018 dataset but *never* appeared on any in the dataset, and at all to the best of our knowledge.

Another trend is the use of digits, especially in the prefix of domain names which is relatively rare for AGD. Also, note that the pykspa domain “abfmfid.net” resulted in the adversarial domain name “poqhfid.net” that involves a consonant-vowel-consonant (CVC) pattern that is common in English readable words, and accordingly in benign domains.

Table 4: Examples of adversarial domain names generated by MaskDGA

DGA	Original	After MaskDGA
qadars	02sygu4egq8m.net	_2_yq-apgq8m.net
fobber	coclocpxjwgoefjih.net	_oclocpxjwfjfh.net
prosilikefan	csdwxmk.biz	lqjw1mk.biz
pykspa	abfmfid.net	poqhfid.net
tinba	bcpprwxhktb.pw	2__rncqxktb.pw

Example of algorithmically generated domains from six DGA families in their original and adversarial form.

6 EVALUATION OF ADVERSARIAL DEFENSES

6.1 Overview

The emergence of adversarial attacks was followed by studies that suggested appropriate defenses. In this section we evaluate the

effectiveness of two common adversarial defenses, namely adversarial re-training and distillation, against the presented MaskDGA evasion technique.

6.2 Tested defenses

The first defense evaluated is adversarial re-training. In adversarial re-training [53] the training set is augmented with adversarial samples so that the resulting model is more robust to adversarial samples. While re-training is considered an effective defense against adversarial samples, it requires prior knowledge of the defender about the attack and it does not necessarily generalize to other attacks [19].

We tested two different adversarial re-training processes: re-training using MaskDGA (denoted as *MaskDGA re-train*) and re-training using DeepDGA (denoted as *DeepDGA re-train*).

In **MaskDGA re-train** each DGA classifier (i.e., the targeted model) is initially trained on the same training set as described in Section 5.2. Then, we randomly selected one hundred AGD names from each of the twenty DGA families (2,000 domains in total) and applied the MaskDGA on them. The targeted model was then re-trained for two additional epochs on a new training set that consists of the original training set and the 2,000 adversarial samples.

In order to apply **DeepDGA re-train**, we first trained the DeepDGA GAN architecture using 500k benign domain names and output 63,500 pseudo-random domain names. As suggested by Anderson *et al.* [3], we extend the training set of the targeted model with these domain names which are labeled as the “benign” class, in order to defend against adversarial attacks.

Distillation networks [33] were proposed as a defense against adversarial perturbation that uses a special parameter called the

“distillation temperature” T . Setting high temperature values (e.g., $T = 10$): for the softmax layer as induces smoother decision bounds as follows:

$$\text{Softmax}(X, T) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

where $\text{Softmax}(X, T)$ is the softmax layer, T is the temperature, and z_i is the i -th logit. After training the initial network with a high distillation temperature, the softmax values obtained (denoted as “soft labels”) are now used to train a new network (denoted as the “distilled network”) which is more resistant to attacks.

As a general adversarial defense, distillation networks have been evaluated on adversarial attacks against the MNIST [26] and CIFAR-10 [24] datasets and for both the effectiveness of the attacks dropped from a 95% of misclassification rate to less than 5%. The main intuition regarding the reduced effectiveness of attacks is that the distilled soft labels allow samples that are in between labels (e.g., a figure of the digit ‘3’ that resembles an ‘8’) to be classified near the decision boundary instead of in the center, which makes the trained model vulnerable to attacks.

Our implementation of the distillation defense relies on the implementation of [10], in which the same temperature was used for both the initial and distilled networks, respectively named teacher and student. Moreover, setting the temperature to values that are larger than 40, as was done on images in [33], failed to converge for DGA detection in which the classes of samples more often overlap. Therefore, we set the temperature to 10.

6.3 Results

Table 5 presents the average F1-score measure of the four models in the cases of *NoAttack*, *DeepDGA* attack, and *MaskDGA* attack, and when no defense is applied (*NoDefense*), when using *MaskDGA* retrain, *DeepDGA* retrain, and *Distillation*.

The results presented in Section 5.6 indicate that, when no defense is applied, the models are vulnerable to domain names that were generated by the DeepDGA attack and even more vulnerable to domain names that were generated by MaskDGA.

The distillation network defense slightly improves the F1-scores against all attacks and across all models but is inefficient overall. The primary reason for the inefficiency of the distillation defense is that it is designed for general misclassification (i.e., misclassification for any other class other than the correct class Y) and not a targeted misclassification (i.e., a misclassification for a specific class Y').

The DeepDGA retrain defense significantly improve the robustness of the targeted models against the DeepDGA attack for three of the four models (all but CMU). The DeepDGA hardening is ineffective against the MaskDGA evasion technique, but it comes with the cost of reducing the effectiveness of detecting non-adversarial AGD names.

The MaskDGA retrain defense provides the highest average F1-score results on all of the attacks and for all targeted models. The effectiveness of detecting non-adversarial AGD names is still reduced in comparison to the targeted models without any defense, but overall it has better results on average, than the alternatives of DeepDGA hardening and distillation networks. The effectiveness of the defense against the DeepDGA attack has an average

F1-score of 0.6675, which is better than the effectiveness of the DeepDGA retrain defense against the MaskDGA evasion technique. This suggests that overall, the defenses do not fully prevent general adversarial attacks against DGA classifiers.

7 PRACTICAL IMPLEMENTATION OF MASKDGA

Resilient communication between bots and their command and control server using a DGA is normally established as follows. The bot operator and the bots have an implementation of the DGA mechanism in their source code and share a secret input seed. Periodically, the botnet operator executes the DGA with the secret input seed to generate a large number of domain names and attempts to register them. The domain names that are successfully registered, are configured by the bot operator to resolve to the command and control system. In turn, the bot executes the DGA with the shared secret input seed to generate the same list of domain names that was generated by the bot operator. The bot iterates the generated domain names and attempts to connect each one of them until it receives a reply from from the command and control server. The bot that executes the DGA might reside on hosts which have limited computational and storage resources (e.g., IoT devices or point-of-sale machines). Therefore, it is important to understand the practical implementation of the presented evasion technique in order to understand its requirements and limitations.

Table 5: Evaluation of the effectiveness of the adversarial defenses (average F1-score).

Defense / Attack	No Attack	DeepDGA	MaskDGA
CNN[Invincea]			
No defense	1.00	0.84	0.39
Distillation	1.00	0.85	0.42
DeepDGA retrain	0.87	0.97	0.40
MaskDGA re-train	0.96	0.72	0.96
LSTM[Endgame]			
No defense	0.98	0.74	0.53
Distillation	0.98	0.74	0.52
DeepDGA retrain	0.96	0.88	0.57
MaskDGA retrain	0.95	0.77	0.91
biLSTM[CMU]			
No defense	0.98	0.78	0.55
Distillation	0.98	0.86	0.51
DeepDGA retrain	0.94	0.48	0.54
MaskDGA retrain	0.93	0.64	0.85
CNN + LSTM[MIT]			
No defense	0.95	0.74	0.51
Distillation	0.58	0.45	0.58
DeepDGA retrain	0.91	0.92	0.49
MaskDGA retrain	0.91	0.54	0.80

Evaluation of the average F1-score of targeted models when applying the distillation, DeepDGA retrain and adversarial retrain defenses.

Table 6: Practical implementation requirements.

Requirement	Memory	Supported platforms
TF Lite	300KB	Android / Embedded / Desktop
Model	10MB	-
Input Repr.	2.4KB / AGD	-
JSM Repr.	38.4KB / AGD	-

The platform and memory requirements of a bot to execute MaskDGA. A bot would load a pre-trained substitute model using TensorFlow Lite that is supported on various environments and represent every AGD input and output JSM representation as a binary character-level matrix in memory.

MaskDGA requires the bot to generate a set of AGD, use a substitute model to perform a feed-forward step and back-propagation to acquire the JSM, and add perturbation based on the JSM. We assume that bots that have previously used DGAs are capable of generating a set of AGD names. Therefore, the main focus of this section the requirements of representing the AGD names as character-level binary matrices, loading a previously trained substitute model, and computing the JSM. The list of requirements is summarized in Table 6.

The longest AGD name in the rich DMD-2018 dataset has 65 characters, and the number of allowed symbols for domain names include case-insensitive English letters, digits, hyphens, underscores and dots, for an overall of 39 symbols. There a binary character-level representation of $65 \cdot 39 = 2535$ entries of bits is sufficient for the representation every single AGD name.

The substitute model that we trained can be serialized to the HDF5 binary serialization format³ and later re-loaded with TensorFlow. Since TensorFlow’s development libraries often exceeds 100 MB, TensorFlow Lite⁴ can be used. TensorFlow Lite weighs less than 300 KB and is designed to load pre-trained models on Android, embedded devices and Desktops and supports various programming languages such as: C, C++, Java and Python. The substitute model that was trained in Section 4.2 had 160,853 parameters and is serialized to a 10MB HDF5 file.

For every AGD name the attacker is required to perform a single feed-forward step and a single back-propagation which is negligible in computation resources (since no training is required). The result of the backpropagation is the JSM matrix of gradients that has a similar number of entries to that of the input representation (2,535), but in contrast, every entry has 16 bits instead of a single bit. Therefore, the overall JSM representation has slightly less than 38.4KB per AGD name.

8 CONCLUSIONS

This study portrays an adversarial machine learning evasion technique on DGA classifiers. The evaluation of the evasion technique on four state of the art, recently published DGA classifiers indicates that their performance degrades significantly. MaskDGA is evaluated along with two other attacks, DeepDGA and random, which also resulted in a degradation in the performance of the

DGA classifiers, although to a lesser extent than that caused by the MaskDGA. The performance degradation caused by the random attack helps us argue that character-level DGA classifiers memorize specific lexicographic patterns of known DGA and thus are extremely vulnerable to adversarial attacks.

The GAN architecture used by DeepDGA (1) requires extensive hyper-parameter tuning, (2) does not easily converge, and therefore is unstable and unexpected, (3) requires an approximation technique to address the discrete values of the input (i.e., the domain name), and (4) was trained to generate domains based on legitimate character distribution, and therefore is likely to generate benign (already registered) domain names. On the other hand, the MaskDGA technique is (1) designed specifically for generating malicious domains (i.e., the substitute model is trained using both legitimate and malicious domains) and is therefore more successful, (2) requires minimal tuning (it only requires training a simple substitute model), and (3) is applied as a wrapper on the original DGA mechanism used by the botnet. Moreover, MaskDGA can be executed by bots with limited resources as long as they install TensorFlow lite (supported on multiple platforms) and load a pre-trained model that weighs less than 10MB, thus making the evasion technique practical for botnets.

Additionally, we tested MaskDGA and the DeepDGA attack against three adversarial defenses, namely: MaskDGA re-training, DeepDGA re-training and distillation networks. We conclude that distillation is ineffective against targeted misclassifications. The DeepDGA retraining was effective against the DeepDGA attack but caused the targeted model to degrade against non-adversarial AGD names. Finally, MaskDGA re-training on the MaskDGA adversarial samples had better results on average, but it still does not provide a perfect defense against other adversarial attacks such as DeepDGA. These results demonstrate the vulnerability of character-level DGA classifiers to adversarial attacks and potential solutions to that are further discussed in Section 10.

9 RELATED WORKS

The concept of using adversarial learning for generating malicious adversarial samples was previously investigated by [20, 38, 39, 44] for generating malware that evade machine learning-based anti-malware software and [14, 17] for evading text classifiers.

DeepDGA [3] extended these studies to the area of DGA detection by proposing a generative adversarial learning application that generates pseudo-random domain names that appear legitimate and thus can be used to augment the benign class in the training set for classification robustness. DeepDGA can be regarded as both an evasion technique that produces pseudo-random domain names and as a defense mechanism for classification robustness. As an evasion technique, DeepDGA is impractical for botnets as the generated domains are short and readable and therefore either expensive or not available for carrying out large attacks by botnets. Moreover, the performance of MaskDGA is superior to that of DeepDGA and simpler to implement and tune, since it is based on a discriminative model which is less complicated computationally than generative models.

Character-level adversarial attacks were previously suggested in HotFlip [16] to cause misclassification of words by text classifiers in

³<https://www.h5py.org/>

⁴<https://www.tensorflow.org/lite/overview>

the white-box adversarial setting by replacing, inserting or deleting a single character. In contrast, the presented evasion technique is applied in a black-box adversarial setting in which a substitute model is trained. Also, deleting or inserting characters is forbidden to avoid the generation of longer domain names which are sensitive to detection and short domain names which are more expensive and less available. In addition, in MaskDGA the replacement of characters is performed on half of the characters, where every character is replaced once, in order to preserve a unique set of generated domains. In contrast, HotFlip always focuses on the character that induces the highest gradient since there's no restriction of replacing multiple words to the same words. For example, the words "open," "pea," and "ten" can all change to the word "pen" in a single flip, but MaskDGA ensures that the chances of outputting the same domain is small, resulting in a larger unique domain names, and accordingly making it difficult to detect all of the generated domain names.

10 DISCUSSION AND FUTURE WORK

Character-level DGA classifiers gained popularity largely due to their simplicity, the availability of training data, and the ability to use them for real-time detection. We therefore aimed at evaluating the robustness of these classifiers against evasion techniques. Since the evaluation of MaskDGA indicates the vulnerability of these classifiers, we propose potential solutions for improved robustness. Firstly, DGA classifiers can improve their robustness against adversarial attacks using additional contextual features (e.g., DNS traffic features and WHOIS features [5, 13, 41]) are also being used for classification. Another approach for robustness should focus on DGA detection systems that would detect domain names with novel lexicographic representation that implies the existence of either a new DGA process, a new secret seed or an adversarial attack.

MaskDGA can be generalized for other security scenarios, as well as non-security scenarios, in which character-level classifiers attempt to predict the malice of a sample. For example, phishing URLs classifiers can re-crafted using the presented technique to evade detection, and malware strings can be automatically replaced to overcome signature-based anti-malware tools. We believe that using MaskDGA for adversarial re-training can improve the robustness of the models in these cases as well.

11 ACKNOWLEDGEMENTS

The authors would like to thank Hyrum Anderson for answering our questions regarding the DeepDGA paper and disclosing the source code of DeepDGA. Additionally, the authors would like to thank R Vinayakumar for sharing the DMD-2018 dataset that was used in this study.

REFERENCES

- [1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, Vol. 16. 265–283.
- [2] Esraa Alomari, Selvakumar Manickam, BB Gupta, Shankar Karuppayah, and Rafeef Alfari. 2012. Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art. *arXiv preprint arXiv:1208.0403* (2012).
- [3] Hyrum S. Anderson, Jonathan Woodbridge, and Bobby Filar. 2016. DeepDGA: Adversarially-Tuned Domain Generation and Detection. 13–21. <https://doi.org/10.1145/2996758.2996767>
- [4] Hyrum S Anderson, Jonathan Woodbridge, and Bobby Filar. 2016. DeepDGA: Adversarially-tuned domain generation and detection. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*. ACM, 13–21.
- [5] Manos Antonakakis, Roberto Perdisci, Yacin Nadjji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware.. In *USENIX security symposium*, Vol. 12.
- [6] Pavol Bielik, Veselin Raychev, and Martin Vechev. 2017. Character Level Based Detection of Dga Domain Names. 1–17.
- [7] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. 2011. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *International workshop on multiple classifier systems*. Springer, 350–359.
- [8] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrđić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 387–402.
- [9] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).
- [10] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.
- [11] François Chollet et al. 2015. Keras. (2015).
- [12] Chhaya Choudhary, Raaghavi Sivaguru, Mayana Pereira, Anderson C Nascimento, and Martine De Cock. [n. d.]. Algorithmically Generated Domain Detection and Malware Family Classification. 1–15. <https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/>,
- [13] Ryan R Curtin, Andrew B Gardner, Slawomir Grzonkowski, Alexey Kleymenov, and Alejandro Mosquera. 2018. Detecting DGA domains with recurrent neural networks and side information. *arXiv preprint arXiv:1810.02023* (2018).
- [14] Prithviraj Dasgupta, Joseph Collins, and Anna Buhman. 2018. Gray-box Techniques for Adversarial Text Generation. 18–19.
- [15] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481* (2016).
- [16] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2014. HotFlip: White-Box Adversarial Examples for Text Classification. (2014).
- [17] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018* (2018), 50–56. <https://doi.org/10.1109/SPW.2018.00016>
- [18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
- [19] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. [n. d.]. Explaining and harnessing adversarial examples. CoRR (2015). (n. d.).
- [20] Weiwei Hu and Ying Tan. 2017. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. (2017). <http://arxiv.org/abs/1702.05983>
- [21] Jonathan Hui. 2018. GAN - Why it is so hard to train Generative Adversarial Networks! (June 2018). https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b
- [22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [23] Joewie J Koh and Barton Rhodes. 2018. Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings. *arXiv preprint arXiv:1811.08705* (2018).
- [24] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR-10 dataset. *online: http://www.cs.toronto.edu/kriz/cifar.html* (2014).
- [25] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. <https://arxiv.org/abs/1611.01236>
- [26] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist> 2 (2010).
- [27] Vysakh S Mohan, R Vinayakumar, KP Soman, and Prabaharan Poornachandran. 2018. Spoof net: Syntactic patterns for identification of ominous online factors. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 258–263.
- [28] Asaf Nadler, Avi Aminov, and Asaf Shabtai. 2019. Detection of malicious and low throughput data exfiltration over the DNS protocol. *Computers & Security* 80 (2019), 36–53.
- [29] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. 2018. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv preprint arXiv:1610.00768* (2018).
- [30] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial

- Samples. <http://arxiv.org/abs/1605.07277>
- [31] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 506–519.
- [32] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 372–387.
- [33] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016* (2016), 582–597. <https://doi.org/10.1109/SP.2016.41>
- [34] Abhinav Pathak, Feng Qian, Y Charlie Hu, Z Morley Mao, and Supranamaya Ranjan. 2009. Botnet spam campaigns can be long lasting: evidence, implications, and analysis. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 37. ACM, 13–24.
- [35] Daniel Plohmann and Gerhards-Padilla Elmar. 2015. DGArchive – A deep dive into domain generating malware. (2015).
- [36] Daniel Plohmann, Khaled Yakdan, Michael Klatt, and Elmar Gerhards-Padilla. 2016. A Comprehensive Measurement Study of Domain Generating Malware. *Proceedings of the 25th USENIX Security Symposium* (2016). <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann>
- [37] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. 2009. Conficker C analysis. *SRI International* (2009).
- [38] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2018. Low Resource Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers. *arXiv preprint arXiv:1804.08778* (2018).
- [39] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. 2018. Generic black-box end-to-end attack against state of the art API call based malware classifiers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11050 LNCS (2018), 490–510. https://doi.org/10.1007/978-3-030-00470-5_{ }23
- [40] Joshua Saxe and Konstantin Berlin. 2017. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. (2017). <http://arxiv.org/abs/1702.08568>
- [41] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. 2014. Phoenix: DGA-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 192–211.
- [42] Raaghavi Sivaguru, Chhaya Choudhary, Bin Yu, Vadym Tymchenko, Anderson Nascimento, and Martine De Cock. [n. d.]. An Evaluation of DGA Classifiers. ([n. d.]).
- [43] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. 2009. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 635–647.
- [44] Octavian Suci, Scott E Coull, and Jeffrey Johns. 2018. Exploring Adversarial Examples in Malware Detection. (2018). <https://doi.org/arXiv:1810.08280v1>
- [45] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
- [46] Duc Tran, Hieu Mac, Van Tong, Hai Anh Tran, and Linh Giang Nguyen. 2018. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing* 275 (2018), 2401–2413. <https://doi.org/10.1016/j.neucom.2017.11.018>
- [47] R Vinayakumar, Prabaharan Poornachandran, and KP Soman. 2018. Scalable Framework for Cyber Threat Situational Awareness Based on Domain Name Systems Data Analysis. In *Big Data in Engineering Applications*. Springer, 113–142.
- [48] R Vinayakumar, Prabaharan Poornachandran, and K P Soman. 2018. *Big Data in Engineering Applications*. Vol. 44. Springer Singapore. <https://doi.org/10.1007/978-981-10-8476-8>
- [49] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. 2018. Detecting malicious domain names using deep learning approaches at scale. *Journal of Intelligent & Fuzzy Systems* 34, 3 (2018), 1355–1367.
- [50] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 1041–1044.
- [51] Wikipedia contributors. 2018. Botnet – Wikipedia, The Free Encyclopedia. (2018). <https://en.wikipedia.org/w/index.php?title=Botnet&oldid=873006619> [Online; accessed 22-December-2018].
- [52] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant. 2016. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. (2016). <http://arxiv.org/abs/1611.00791>
- [53] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. [n. d.]. Adversarial Examples : Attacks and Defenses for Deep Learning. ([n. d.]), 1–20.