IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 22, 2020, accepted August 28, 2020, date of current version September 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020964

MaskDGA: An Evasion Attack Against DGA Classifiers and Adversarial Defenses

LIOR SIDI, ASAF NADLER, AND ASAF SHABTAI
Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel
Corresponding author: Asaf Shabtai (shabtaia@bgu.ac.il)

This work was supported in part by the CONCORDIA project that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant 830927.

ABSTRACT Domain generation algorithms (DGAs) are commonly used by botnets to generate domain names that bots can use to establish communication channels with their command and control servers. Recent publications presented deep learning classifiers that detect algorithmically generated domain (AGD) names in real time with high accuracy and thus significantly reduce the effectiveness of DGAs for botnet communication. In this paper, we present MaskDGA, an evasion technique that uses adversarial learning to modify AGD names in order to evade inline DGA classifiers, without the need for the attacker to possess any knowledge about the DGA classifier's architecture or parameters. MaskDGA was evaluated on four state-of-the-art DGA classifiers and outperformed the recently proposed CharBot and DeepDGA evasion techniques. We also evaluated MaskDGA on enhanced versions of the same classifiers equipped with common adversarial defenses (distillation and adversarial retraining). While the results show that adversarial retraining has some limited effectiveness against the evasion technique, it is clear that a more resilient detection mechanism is required. We also propose an extension to MaskDGA that allows an attacker to omit a subset of the modified AGD names based on the classification results of the attacker's trained model, in order to achieve a desired evasion rate.

INDEX TERMS Adversarial learning, deep learning, botnets, DGA.

I. INTRODUCTION

Botnets are groups of interconnected devices that are designed to carry out large-scale cyber-attacks, such as distributed denial-of-service attacks [2], data theft [20], and spam [24]. Modern botnets often involve advanced techniques to prolong their operations by making them less detectable. A domain generation algorithm (DGA) is one such technique which has been used by more than 40 documented botnets in the last decade [27]. DGAs are used to generate a large number of pseudorandom domain names based on a secret input (seed). A bot and its command and control (C&C) server that wish to communicate will execute the DGA with a shared seed in order to generate a sequence of domain names and identify one through which their communication will take place. DGAs can be used to generate thousands of domain names per day which must be identified and analyzed by DGA detection systems in order to shut down the botnet.

Originally, the detection of algorithmically generated domain (AGD) names focused on capturing binary samples of bots, extracting their algorithms and seeds, and generating the domain names in advance for the purpose of mitigation [26]. However, it was found that by using new input seeds, this approach can be easily evaded by botnets. In fact, between 2017 and 2018 at least 150 new seeds were introduced by botnets,[1] a figure which is more than twice the number of documented DGAs, thus demonstrating the ineffectiveness of mitigation based on previously extracted seeds.

An alternative and more generalized approach is to apply an offline data analysis of DNS traffic logs enriched with various features from independent sources of contextual information (e.g., WHOIS data, IP-based information) [38]. The offline data analysis approach provides accurate and robust results, since it often relies on multiple independent data sources. However, this approach cannot be applied as a real-time solution due to the amount of time it takes to collect and process the contextual data collection and processing [34].

The inline mitigation approach to DGA detection can be run in real-time, in contrast to the offline analysis approach, by using machine learning to classify a domain name that

The associate editor coordinating the review of this manuscript and approving it for publication was Alberto Cano.

[1] Based on the changelog of https://data.netlab.360.com/dga/

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

appears in traffic based *solely* on the observed domain name. With the emergence of deep neural networks, the results of inline DGA mitigation have provided high accuracy [28] as well as reduction in real-time latency which is essential for network perimeters that cannot tolerate botnet communication, such as Internet service providers or point of sale networks. Accordingly, inline DGA detection systems has become the topic of extensive academic research, and it resulted thus far in DGA classifiers that identify algorithmically generated domain (AGD) names in real-time (inline) and with high accuracy using deep convolutional neural network architectures [6], [14], [37], [39], recurrent neural network architectures [18], [30], [33] and other architectures [16], [31], [35].

Botnet operators (henceforth referred to as the adversaries) can evade inline DGA classifiers by modifying AGD names to be less detectable to DGA classifiers. This idea was demonstrated by Anderson *et al.* [3] who proposed the DeepDGA technique and Peck *et al.* [25] who suggested the CharBot technique. DeepDGA uses a GAN architecture to generate domain names that appear benign, and CharBot achieves a similar goal by modifying two random characters of existing benign domain names. While both DeepDGA and CharBot can degrade the accuracy of DGA classifiers, they do not leverage adversarial learning that was demonstrated in the past as a powerful attack against neural network architectures that are commonly used by DGA classifiers (i.e., systematically identifying the weaknesses of the classification model). Therefore, the primary goal of our study was to evaluate the use of adversarial learning to evade DGA classifiers in comparison to DeepDGA and CharBot's ability to do so, as well as to evaluate adversarial learnings robustness to adversarial defenses.

In this paper, we present MaskDGA, an adversarial learning technique that modifies AGD names so they are misclassified as benign by DGA classifiers. MaskDGA is applicable within the adversarial training capability threat model (Section II) in which an adversary has access to public datasets of AGDs to train a private model, but does not possess knowledge about the architecture or parameters of target models (i.e., DGA classifiers) and cannot acquire their outputs. MaskDGA is comprised of a setup phase, in which the adversary generates the privately trained substitute model, and an attack phase, in which the substitute model is applied to select character modifications to AGD names, such that the modified AGD is misclassified by the substitute model. Based on the concept of adversarial transferability [22], adversarial attacks on the trained substitute model will successfully transfer to other models, regardless of their architecture, as long as they were trained on datasets with a similar distribution and for a similar objective, as in this case of AGD name classification.

The evaluation of MaskDGA was performed using the recently published DMD-2018 dataset of AGD names and four state-of-the-art DGA classifiers [8]. DeepDGA [3] and CharBot [25] were used as baseline attacks, and MaskDGA was shown to outperform them on all of the four target models. We also evaluated MaskDGA against the same DGA classifiers enhanced with adversarial learning defenses, namely distillation [23] and adversarial retraining on each adversarial attack, and assessed their effectiveness. The results show that adversarial retraining has some limited effectiveness against the evasion technique, but that a more resilient detection mechanism is required.

MaskDGA has an additional unique property compared to other evasion techniques — a controllable rate of evasion. The substitute model provides every modified AGD with a probability score ranging from zero (benign with high certainty) to one (AGD with high certainty). An adversary can define a threshold for the probability score and produce modified AGD names until reaching the desired number of domain names for whom the probability score exceeds the defined threshold. This property is useful for an adversary that wishes to completely evade detection in sensitive networks (i.e., not a single domain is detected) and/or force a DGA detection system to an unacceptable rate of false positives in order to identify the more evasive AGD names. Further details and an evaluation of this property appear in Section VI.

The contributions of this paper are as follows:

1) A practical evasion technique for DGA detection machine learning models, which outperforms existing evasion techniques, with a controllable evasion rate.
2) An evaluation of DGA evasion techniques (including MaskDGA) using four state-of-the-art DGA classifiers.
3) An evaluation that assesses the robustness of existing adversarial attacks (including MaskDGA) against commonly used adversarial defense techniques.

II. THREAT MODEL

Under the threat model employed in this study, the objective of the adversary is to modify a set (sequence) of AGD names so that the rate of detection by DGA classifiers is minimized taking into account the following two limitations.

A. TRAINING CAPABILITY

The threat model for all evaluations in this study is the training data capability setting described in [21]. In this setting, we assume that the adversary is able to collect a substitute dataset, sampled from the same distribution used to train the DGA detection system, including its underlying target model. Conversely, the adversary does not possess knowledge of the target model network architecture and cannot collect sample pairs of input and output related to the target model. The training data capability setting reflects real-world adversaries that have access to publicly available AGD datasets but do not have access to DGA detection systems to learn about their architecture and classification outputs.

B. LIMITED NUMBER OF CHARACTER MODIFICATIONS

The adversary can change at most half of the characters for every input AGD name. We expect that changing up to half

of the characters of a long and unreadable AGD name would preserve its unreadability and accordingly is likely to be an unregistered domain name that can be used by a botnet. This setting challenges the adversary to carefully select character modifications that result in unreadable yet evasive AGD names instead of completely replacing a given AGD name with a benign domain name that is already registered and thus cannot be utilized for a botnet operation.

For example, an adversary that wishes to modify the AGD name ''abcdefghijklm.com'' to evade detection can change up to six characters, since the second-level domain (''abcdefghijklm'') consists of thirteen characters. Without this limitation, the adversary could have replaced the AGD name with another name of the same length such as ''stackoverflow.com'' which is a benign and registered domain name that might have appeared in the training set.

III. The EVASION TECHNIQUE

The proposed evasion technique is comprised of a setup phase and an attack phase. In the setup phase, the botnet operator trains a substitute model of a binary DGA classifier and deploys the trained model to every bot. In the attack phase, the adversary and its controlled bots use the trained substitute model to generate domain names through which they will be able to communicate with the C&C while evading detection on the security perimeter. The setup phase is applied just once, after which multiple attack phases can take place.

A. SETUP PHASE

The goal of the setup phase is to produce the substitute model — a DGA classifier trained by the adversary. Later on, the adversary will apply the substitute model on AGD names to gain insights about how to modify the characters of AGD names so they become less detectable by the substitute model. Based on the concept of adversarial transferability [21], the modified AGD names will also become less detectable by the target models. The substitute model is a neural network binary DGA classifier with two requirements:

1) The architecture must have an input one-hot encoding layer and support a differentiable transformation through all of its layers in order to compute the saliency of every possible symbol in every possible location index on a misclassification (explained in detail below).
2) The training set for the substitute model involves adversarial examples alongside normal examples of domain names to improve the robustness of the substitute model (also known as Jacobian-based dataset augmentation [22]).

1) CONSTRUCTING AN INITIAL DGA CLASSIFIER

Initially, the adversary selects a neural network architecture for the substitute DGA classifier, based on any known architecture that was proven successful for DGA classification [18], [30], [33]. If the selected architecture has an embedding layer rather than a one-hot encoding layer as its input layer, it must be replaced with a one-hot encoding layer to meet the first requirement.

The one-hot encoding layer added has two dimensions and a constant size; it has exactly 63 columns to fit the longest possible domain label name based on RFC 1035 [19], 38 rows, corresponding to the valid symbols of domain names, and an additional special pad symbol. Domains that have less than 63 characters are left-padded with the special pad symbol. Formally, the one-hot encoding layer represents an input word W (i.e., a domain name) that is comprised of symbols from the set V, using a binary matrix $X_{|W|\cdot|V|}$ such that

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

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.

After selecting the architecture for the substitute model, the adversary trains the substitute model using publicly available datasets of AGD names and benign domain names. The output model is now capable of distinguishing AGD names from benign domain names and supporting gradient computation on the input, but it does not yet meet the second requirement — robustness against adversarial examples, which is obtained by Jacobian-based dataset augmentation.

2) JACOBIAN-BASED DATASET AUGMENTATION

Jacobian-based dataset augmentation [22] refers to the training of a substitute model on adversarial samples along with normal samples to improve the model's robustness. In this step, the adversary iteratively generates adversarial domain names, adds them to the dataset of AGD names and benign domain names, and trains a new substitute model. The adversary generates adversarial domain names by applying the attack phase (Subsection III-B) and adds the adversarial examples to the dataset at a rate such that the newly formed dataset will contain at most 10% malicious labeled domain names, in order to prevent the classifier from forgetting patterns of original AGD names. The dataset is then split into a training and test set. At the end of every dataset augmentation iteration, the adversary discards the previously trained model and saves the a newly trained one. The adversary repeats this step of generating new adversarial domain names based on the test set until the results of the classifier reach convergence (as shown in Figure 1).

3) MODEL DEPLOYMENT

In this step, the most recent substitute model is deployed to every bot controlled by the adversary, and used to generate adversarial domain names, as explained in Subsection III-B.

B. ATTACK PHASE

In the attack phase, the adversary and bots use the trained substitute model to generate domain names through which

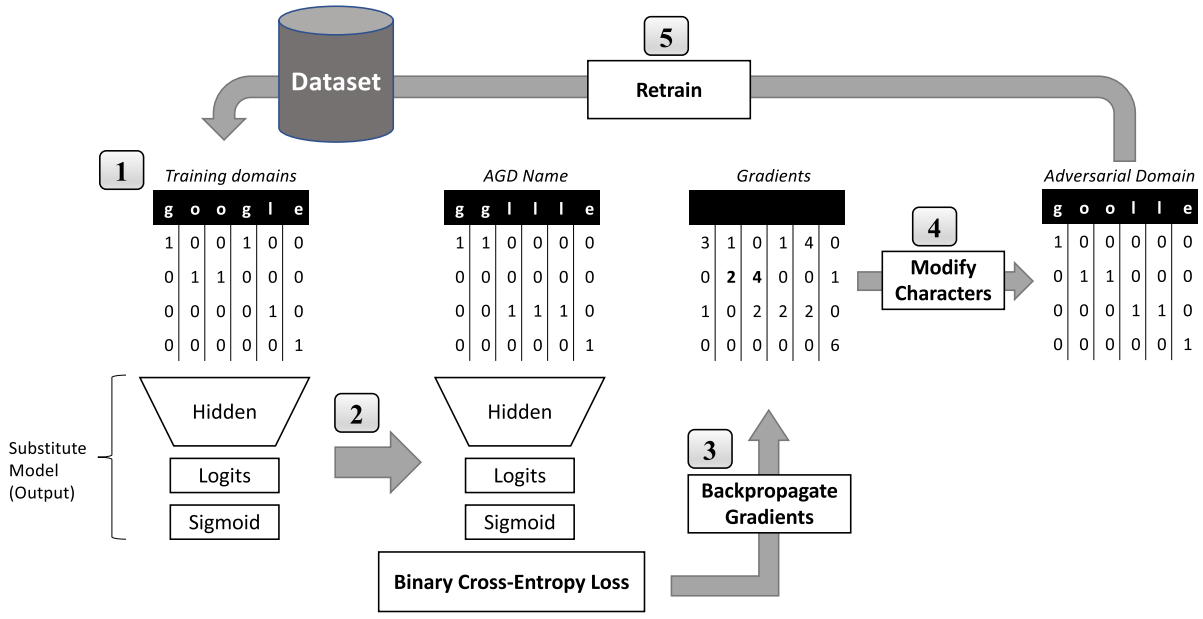


FIGURE 1. The setup phase of MaskDGA starts with the adversary training a binary classifier on a public AGD dataset (1). Then, the adversary applies the attack to generate adversarial examples (2-4), adds them to the dataset (5), and trains a new model until convergence.

they will be able to communicate safely while evading detection on the security perimeter. The attack phase consists of executing a DGA and modifying its resulted AGD names to be misclassified by a target model, thus evading detection. Since a large number of bots already use DGAs [27], MaskDGA is designed *to be applied on the output of the botnet's existing DGA* to turn AGD names into adversarial domain names.

The inputs for the attack phase are the substitute model that was trained in the setup phase and an AGD name, and the output is a modified (adversarial) AGD name that is capable of evading detection. Therefore, the process of selecting the modification is deterministic i.e., applying a specific substitute model to a specific AGD name will always yield the same proposed modifications.

1) COMPUTING THE TARGET LOSS AND GRADIENTS

The adversary computes the output of the pre-trained substitute model on the input AGD name and evaluates the *target loss*. The target loss in our case is the binary cross-entropy loss function between the output of the substitute model for an AGD name X and Y^*, the benign class label. A formal notation of the target loss appears in Equation 1.

$$\mathcal{L}(F(X), Y^*). \tag{1}$$

Then, the backpropagation algorithm is applied on the substitute model to compute the partial derivatives of the target loss with regard to each of the input features of X. The partial derivative (gradient) value for each input feature $X_{i,j}$ indicates its saliency in the misclassification of X as a benign domain (as portrayed in Equation 2).

$$S(X)_{i,j} = \frac{\partial \mathcal{L}(F(X), Y^*)}{\partial X_{i,j}}. \tag{2}$$

The set of gradients for every input feature of an AGD name directs the adversary to replace existing symbols in the specific location indices with new symbols that have higher gradient values and accordingly are more likely to result in a misclassification of the output domain as benign. Formally, we denote $S(X)$ as the Jacobian saliency matrix that maps every feature of X to its saliency, as shown in Equation 3:

$$S(X) = \begin{bmatrix} S(X)_{0,0} & \cdots & S(X)_{|W|-1,0} \\ \vdots & \ddots & \vdots \\ S(X)_{0,|V|-1} & \cdots & S(X)_{|W|-1,|V|-1} \end{bmatrix} \tag{3}$$

where $|W| = 63$ is the maximal number of characters, and $|V| = 38$ is the number of symbols, as explained in the setup phase (Subsection III-A).

2) MODIFICATION OF THE AGD NAME

The adversary selects the character replacement for the input domain name X based on the saliency matrix $S(X)$. As a first step, the adversary must select the character positions in which the symbol would be replaced. This is achieved by scoring the character positions based on their importance for misclassification. The scoring of character positions (columns of $S(X)$) is determined based on their maximal gradient value (row of $S(X)$) based on the definition in Equation 4.

$$\forall i \in [|W|] : PositionScore(X, i) = \max_{0 \le j < |V|} S(X)_{i,j} \tag{4}$$

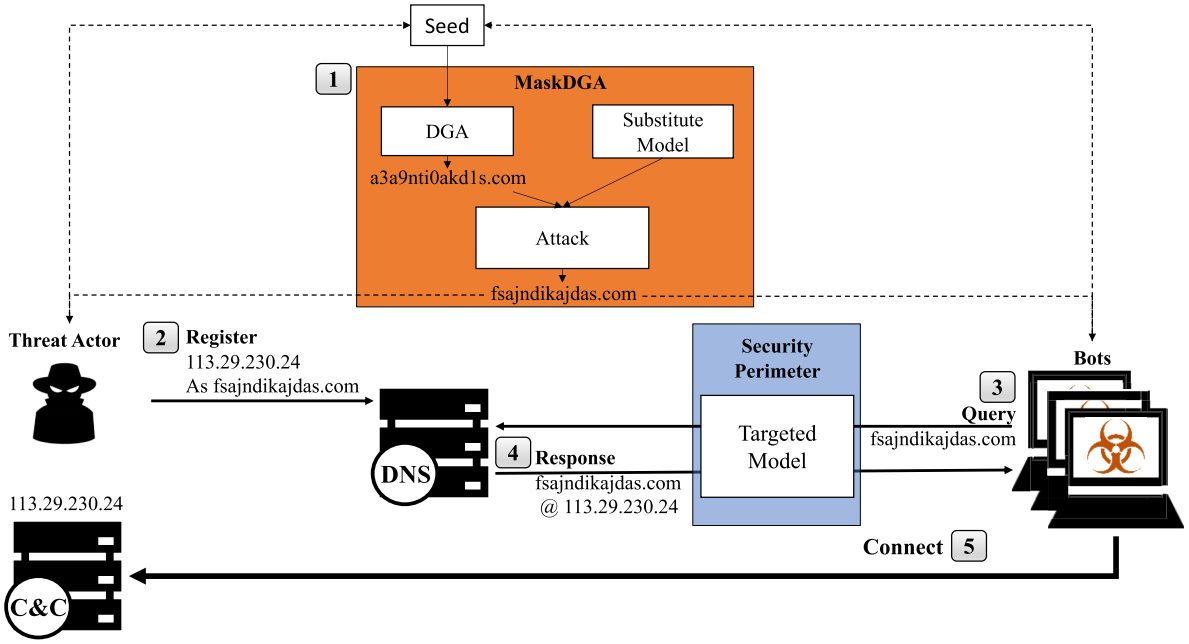


FIGURE 2. The attack phase starts with the adversary and its controlled bots executing a DGA over a shared secret input seed to produce AGD names, and then use the substitute model to modify the AGD names into an adversarial domain names (1). The adversary and the bots select the domain name through which they will communicate while evading detection on the security perimeter (2-5).

After providing every character position with a score, the adversary replaces the symbol only in character positions that exceeds the median score over all position scores. The choice of the median score leads (by definition) to having at most half of the symbols being replaced. The new symbol that will be placed in the location index i is determined by the row with the highest gradient value in column i, as specified in Equation 5:

$$NewSymbol(X, i) = \underset{0 \le j < |V|}{\operatorname{argmax}} S(X)_{i,j} \tag{5}$$

If more than one symbol value is assigned with the maximal value, the first symbol value (by alphabetical order) for consistency. A visual example of the character modification is provided as part of the setup phase in Figure 1. Additionally, a demonstration of the attack phase using Python code is available on Section X of the Appendix.

3) C&C COMMUNICATION

The adversary and its bots share the same model and seed to generate domain names and communicate similarly to existing DGA schemes (see illustration on Figure 2). The deployed substitute model is deterministic and thus guarantees the same adversarial domain names for the adversary and its controlled bots.

IV. EVALUATION

In this section, we present the experiments conducted in order to evaluate the MaskDGA evasion technique. MaskDGA was tested on four recently proposed models which are based on deep learning DGA classifiers, using a public dataset of AGD names from twenty DGA families. An additional test was conducted to evaluate the efficacy of MaskDGA on each specific DGA family.

The performance of MaskDGA was compared against two baseline attacks: DeepDGA [3] and CharBot [25]. The code for DeepDGA appears on GitHub.[2] with a similar configuration to that reported in the paper. Additionally, we implemented the CharBot technique based on the paper. Lastly, we evaluated countermeasures against adversarial DGA attacks by testing the target models against the same attacks but with the addition of two adversarial defenses: adversarial retraining [36] and distillation networks [23]. Note that one of the main differences between the baseline attacks and MaskDGA, is that the baseline attacks use benign domains in order to generate new AGD names, while MaskDGA modifies given (existing) AGD. The target models evaluated are based on the following state-of-the-art DGA classifiers:

1) **Endgame** [33], which is based on an LSTM network (denoted as LSTM [Endgame]). The model architecture consists of an input embedding layer (128 units), an LSTM layer (128 units), a dropout layer, and an output dense layer (one unit) with a sigmoid activation.
2) **CMU** [10], which is based on a bidirectional LSTM network (denoted as biLSTM [CMU]). The model architecture consists of an input embedding layer (128 units), a bidirectional LSTM layer (64 units),

[2]https://github.com/roreagan/DeepDGA/

and an output dense layer (one unit) with a sigmoid activation.

3) **Invincea** [15], which is based on a parallel CNN network (denoted as CNN [Invincea]). The model architecture consists of an input embedding layer (128 units), an array of four CNN filters (kernels) of different sizes. Overall, there are 256 filters of sizes 2,3,4 and 5, and 128 units. The filters are concatenated to a dense layer, which is followed by two dense layers (1,024 units) and an output dense layer (one unit) with a sigmoid activation.
4) **MIT** [32], which is based on stacked CNN and LSTM networks (denoted as CNN + LSTM [MIT]). The model architecture consists of an input embedding layer (128 units), a 1D CNN layer of 128 filters with a kernel of size three, an LSTM layer (64 units), and an output dense layer (one unit) with a sigmoid activation.

The target models were implemented using the source code published by [4]. All of the models were trained and optimized with the Adam optimizer for five epochs, in all cases achieving an F1 score of at least 0.96 on the test set of benign and original AGD names (see Table 2).

The substitute model used in our evaluation is based on the CNN architecture of the Invincea model (see [15]). The architecture included a 63×38 one-hot encoding layer, an array of four CNN filters with kernels ranging in size from two to five, 256 filters, and 128 units that are concatenated to a dense layer which is followed by two dense layers (1,024 units) and an output dense layer (one unit) with a sigmoid activation. The model was trained for 25 epochs (although on a smaller training set than the target models were trained on) until the validation loss changes by less than 0.001 for five consecutive epochs. The CleverHans library[3] was used to compute the gradients based on the target loss. We evaluated the domain names that were modified in the third iteration of the setup phase.

A. DATASETS

The main dataset used for the training and evaluation is the DMD-2018 dataset [31]. The dataset records includes 100,000 registered, benign domain names, and 297,777 AGD names that were evenly produced from 20 DGA families that were used by bots and is thus practical for both research and real-world applications. The cleaning of dataset was based on removing domain names that albeit being produced by a real DGA cannot be registered for use by adversaries because they are invalid according to RFC 1035 [19]; this requires that each domain has a label with at most 63 characters that consists only of English letters, digits, or hyphens. A detailed explanation of the datasets used is provided below, and a summary appears in Table 1.

1) SUBSTITUTE MODEL DATASET

The substitute model is trained on 23,120 benign domain names and 68,846 original AGD names. The model is tested on a dataset of 5,780 benign domain names and 17,211 original AGD names.

[3]https://github.com/tensorflow/cleverhans

TABLE 1. Datasets.

	Train	Validation	Test
Substitute Model			
Benign	23,120	-	5,780
AGD	68,846	-	17,211
Target Models			
Benign	40,200	13,400	13,400
AGD	142,132	47,377	10,000
CharBot attack	-	-	10,000
DeepDGA attack	-	-	10,000
MaskDGA attack	-	-	10,000

TABLE 2. Evaluation results.

Target model	Precision	Recall	F1 score
CNN [Invincea]			
No Attack	0.93	0.98	0.96
CharBot	0.78	0.62	0.66
DeepDGA	0.89	0.82	0.85
MaskDGA	**0.51**	**0.51**	**0.50**
LSTM [Endgame]			
No Attack	0.96	0.98	0.97
CharBot	0.82	0.59	0.62
DeepDGA	0.90	0.71	0.77
MaskDGA	**0.51**	**0.51**	**0.51**
biLSTM [CMU]			
No Attack	0.95	0.97	0.96
CharBot	0.79	0.60	0.63
DeepDGA	0.91	0.80	0.84
MaskDGA	**0.51**	**0.51**	**0.51**
CNN + LSTM [MIT]			
No Attack	0.94	0.98	0.96
CharBot	0.78	0.62	0.66
DeepDGA	0.89	0.82	0.85
MaskDGA	**0.51**	**0.51**	**0.50**

2) TARGET MODELS' DATASET

The four target models were trained on 40,200 benign domain names and 142,132 original AGD names which is nearly twice the size of the training set for the substitute model, thus providing every target model with an advantage against the attacks. An additional validation set was used for early stopping to avoid overfitting. The validation set included 13,400 benign domain names and 47,377 original AGD names. Each target model was evaluated separately against a test set comprised of 13,400 benign domains and 10,000 AGD domain names that were generated by original DGAs, DeepDGA, CharBot, and MaskDGA.

B. RESULTS

The metrics used to evaluate the effectiveness of the evasion technique on the target models are precision, recall, and the F1 score, as commonly used for DGA classification evaluation. These metrics are listed on Table 2 for each target model on the test set of original AGD names (No

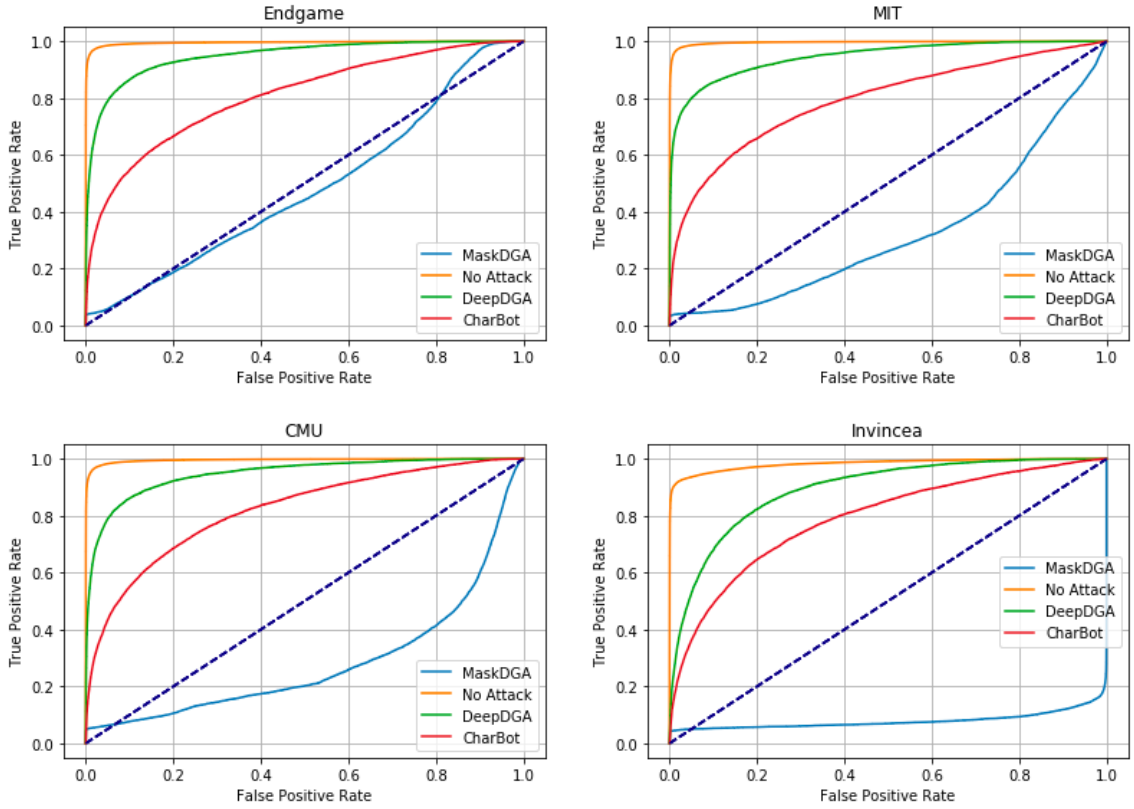


FIGURE 3. The receiver operating characteristic (ROC) curve of the four target models on the test sets.

Attack), the CharBot and DeepDGA baseline attacks, and the MaskDGA evasion technique.

1) EVALUATION AGAINST BASELINE ATTACKS

All of the target models demonstrated an almost perfect classification of AGD names (No Attack) with a small false positive rate (FPR) as expected, thus validating the correctness of the models' training process (as shown in Figure 3). The originally reported results of Charbot (FPR=1% / TPR=15.5%) [25] and DeepDGA (FPR=1% / TPR=45.5%) [3] match our reported results. MaskDGA (FPR=1% / TPR=5%) outperforms both CharBot and DeepDGA (as seen on Figure 4) for every acceptable FPR thus demonstrating the benefit of adversarial learning in the construction of an effective evasion technique.

2) EVALUATION PER DGA FAMILY

We evaluate the effectiveness of the evasion technique on specific DGA families which are characterized by different lengths and use of characters of their resulted AGD names. Table 3 presents the F1 score for the 20 DGA families that are present in the test set.[4]

[4]Note that the DeepDGA and CharBot techniques are not included in Table 3, since they are not applied to existing DGA families and are therefore not relevant for this specific evaluation.

As can be seen in the results, MaskDGA performs well with all DGA families. We analyze the results of MaskDGA on the DGA families on which MaskDGA performed the best and the worst to obtain key insights. The DGA families on which MaskDGA performed best (lowest F1 score) are the Suppobox and Tinba DGA families. The Suppobox and Tinba DGAs use short (on average) domain names (see Table 3). Also, the Suppobox DGA is a dictionary-based DGA, i.e., it concatenates random English dictionary words to generate domain names (for instance, arivenice.ru is a domain name that is generated by Suppobox, and oykjietwrmlw.ru is a domain name generated by Tinba).

In contrast, the DGA families for which MaskDGA performed worst were CoreBot and NewGOZ. CoreBot is the DGA that produces the longest AGD names of the DGA families in our dataset, with up to 28 characters (see Table 3). Both CoreBot and NewGOZ are hash-based DGAs, i.e., they use hash functions to generate domain names (for instance, wxuhq8wvs26lgw18ctyvukx.ddns.net is a domain name that is generated by CoreBot, and 1e0doiiv1oka5c1eykwsm15v3ict.com is a domain name generated by NewGOZ).

Therefore, an additional interpretation of the results is that MaskDGA performs better when provided with AGD names that are more readable (e.g., sampled from a dictionary, as in Suppobox) and names that are, on average, shorter. A similar

TABLE 3. Evaluation Results per DGA family (average F1 score).

Model/DGA	banjori	corebot	dircrypt	dnschanger	fobber	murofet	necurs	newgoz	padcrypt	proslikefan	pykspa	qadars	qakbot	ramdo	ranbyus	simda	Suppobox	symmi	tempedreve	tinba
Min Domain Length	11	12	20	10	10	8	7	20	16	6	6	12	8	16	14	5	8	8	7	12
Max Domain Length	26	23	20	10	17	16	21	28	16	13	15	12	25	16	14	11	26	15	11	12
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
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
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
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
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

observation was made in [7] which indicates than shorter AGD names are more difficult to detect, but our results also show that with a limited number of character replacements, DGAs that produce shorter (on average) and readable dictionary words are preferable for MaskDGA and enable it to better evade DGA detection.

3) EXAMPLES OF MODIFIED DOMAIN NAMES

Table 4 presents nine examples of adversarial domain names that were generated by MaskDGA. The observations that can be made based on these examples fall into three categories:

1) Use of digits as separators for long words (case 1). Since AGD names are longer, on average, than benign domain names, DGA classifiers tend to identify longer domain names as AGD names. However, we observe that replacing English characters with digits in long domain names causes the DGA classifier to misclassify the adversarial domain names as benign.
2) Use of hyphenation to separate for long words (case 2). Similar to case 1, replacing English characters with hyphens also causes misclassification by DGA classifiers.
3) Use of vowels in short words (case 3). Benign domain names are often shorter and more readable than AGD names. The lack of vowels between consonants is often a signal for words that are difficult to read, and accordingly we observe that MaskDGA attempts to modify characters so that a vowel would more frequently appear between two consonants.

C. MaskDGA PRACTICALITY

MaskDGA's attack phase requires bots to load the substitute model that was trained in the setup phase (Subsection III-A) and use it to modify domain names (as explained in Subsection III-B). The substitute model that we evaluated is based on the Invincea model [15]) and has 154,307 parameters. When serialized to an HDF5 binary file format [9], the model size is 684KB, which is ten times smaller than the DeepDGA model whose size is 6.53MB [25], and five times larger than the CharBot model which requires 162KB [25]. The MaskDGA substitute model can be loaded using the TensorFlow Lite library [1] which requires less than 300KB of storage and is designed to be executed on machines with low computational resources (such as bots).

TABLE 4. Examples of MaskDGA AGD names.

Case	Original	After MaskDGA
1	wauyfcpjka.com cmhbbnbhpsbzrr.net onytbjjrbjyu.ru	n9347cpjka.com 9bm4u3ahpsbzrr.net 0r070njrbjyu.ru
2	hmuyifcfwqmcri.com ehharncjtuxdfw.org qkgsssqqcksuwuyo.org	2115-7-fwqmcri.com 19–162jtuxdfw.org 1-11625qckspwuyo.org
3	relmxt.ru nkwchb.info amwrwl.biz	eelix8.ru uihchb.info slerwl.biz

The complexity of applying MaskDGA to generate a single domain name is equivalent to the complexity of a single back-propagation step on the substitute model. On a machine with limited resources, the modification of a single domain name is expected to take less than a second. Therefore, an adversary that uses a DGA that produces hundreds of domains per day will be able to modify them in less than several seconds per day. The consumption of MaskDGA is more intensive than CharBot, but it is sufficiently practical for the generation of thousands of domain names on a daily basis, even on a machine with limited computational resources.

V. EVALUATION OF ADVERSARIAL DEFENSES

We evaluate the effectiveness of two common adversarial defenses (adversarial retraining and distillation) against the proposed MaskDGA evasion technique.

A. TESTED DEFENSES

1) DISTILLATION NETWORKS

Proposed as a defense against adversarial examples, distillation networks [23] attempt to make the decision bounds of the latent feature space more smooth and accordingly reduce the effectiveness of adversarial examples that lie on

the decision bounds. Distillation networks are designed for multiclass classification tasks in which a neural network architecture uses a softmax output layer. The softmax output layer is affected by a special parameter called the "distillation temperature" which controls the level of smoothness, such that after training a neural network with a high temperature, the softmax values obtained (denoted as "soft labels") are used to train a new network (denoted as the "distilled network") which is more resilient to attacks. Our implementation of the distillation defense relies on the implementation of [5], in which the same temperature was used for both the initial and distilled networks, respectively referred to as the teacher and the student. Since our evaluation is based on a binary classification task rather than a multiclass task, we replace the binary output layer of our architecture with an equivalent binary softmax layer so that distillation would be applicable.

2) ADVERSARIAL RETRAINING

Adversarial retraining [36] is applied by augmenting the training set with adversarial examples so that the resulted model becomes robust when classifying new adversarial examples. While retraining 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 [13]. Phrased differently, evaluating a target model enhanced with adversarial retraining measures MaskDGA's efficacy in an arms race between an aware adversary and an aware defender (i.e., a DGA detection system).

We tested three different adversarial retraining processes: retraining using the MaskDGA, DeepDGA, and CharBot attacks. For each of the evasion techniques, we augmented the training set of the target model with 2,000 adversarial domain names that were generated by that specific evasion technique. Then, we test the target model with the adversarial retraining defense on a test set that includes 10,000 new adversarial domain names of each evasion technique.

B. RESULTS

Table 5 presents the average F1 score of the four models on the cases of No Attack (original AGD), the DeepDGA, CharBot, and MaskDGA attacks with and without adversarial defenses. The possible defense include the cases of no defense, MaskDGA-retrain, DeepDGA-retrain, CharBot-retrain, and distillation.

1) DISTILLATION NETWORK RESULTS

The effect of the distillation network defenses is inconclusive as they occasionally improve or degrade the detection of adversarial examples and are thus an unreliable defense. Given the observed inefficiency of the distillation defense which is designed for general multiclass misclassification (i.e., misclassification for any other class other than the correct class), as opposed to a targeted misclassification (i.e., misclassification as a specific class).

TABLE 5. Evaluation of the effectiveness of the adversarial defenses (average F1 score).

Defense / Attack	No Attack	CharBot	DeepDGA	MaskDGA
CNN [Invincea]				
No defense	0.96	0.66	0.85	0.50
Distillation	0.96	0.66	0.79	0.49
CharBot retrain	0.93	0.64	0.68	0.50
DeepDGA retrain	0.92	0.60	0.97	0.51
MaskDGA retrain	0.92	0.62	0.72	0.95
LSTM [Endgame]				
No defense	0.97	0.62	0.77	0.51
Distillation	0.95	0.61	0.75	0.50
CharBot retrain	0.95	0.65	0.82	0.51
DeepDGA retrain	0.95	0.59	0.99	0.51
MaskDGA retrain	0.93	0.57	0.79	0.98
biLSTM [CMU]				
No defense	0.96	0.63	0.84	0.51
Distillation	0.95	0.64	0.74	0.50
CharBot retrain	0.95	0.63	0.87	0.52
DeepDGA retrain	0.93	0.56	0.99	0.51
MaskDGA retrain	0.93	0.58	0.76	0.98
CNN + LSTM [MIT]				
No defense	0.96	0.66	0.85	0.50
Distillation	0.95	0.61	0.80	0.50
CharBot retrain	0.96	0.67	0.89	0.50
DeepDGA retrain	0.95	0.58	0.98	0.51
MaskDGA retrain	0.95	0.62	0.89	0.97

2) ADVERSARIAL RETRAINING RESULTS

The adversarial retraining defenses improve the accuracy of the target model when classifying the *specific* attack on which it was retrained. However, the accuracy of adversarially retrained DGA classifiers does not improve for *general* cases of evasion techniques on which the classifier was not retrained. This observation leads to the conclusion that adversarial retraining is only helpful against adversarial attacks that appeared in the training set (e.g., such as an arms race between an aware adversary and a DGA detection system), but not against unknown attacks.

VI. GOING UNDER THE RADAR

DGA detection systems are tuned by their administrators to acheive a desired trade-off between the rate of true detections and an acceptable rate of false alarms that can be further investigated by security analysts. In this section, we explore a unique property of MaskDGA that allows adversaries to either completely evade detection by a DGA detection system with a low acceptable rate of false alarms or force an unacceptable rate of false alarms in order to identify the produced and modified AGD names. Recall that the substitute model of MaskDGA and the targeted models are essentially binary DGA classifiers that provide every input domain name with a score representing its probability of being algorithmically generated (henceforth referred to as the model score). Based on the property of adversarial transferability, we anticipate that MaskDGA's substitute model's score would be strongly correlated with that of the targeted models' scores, because they are trained for similar tasks. To this end, we define a parameter called the *drop rate*.

The drop rate parameter defines the rate of adversarial examples to be discarded by the adversary based on the

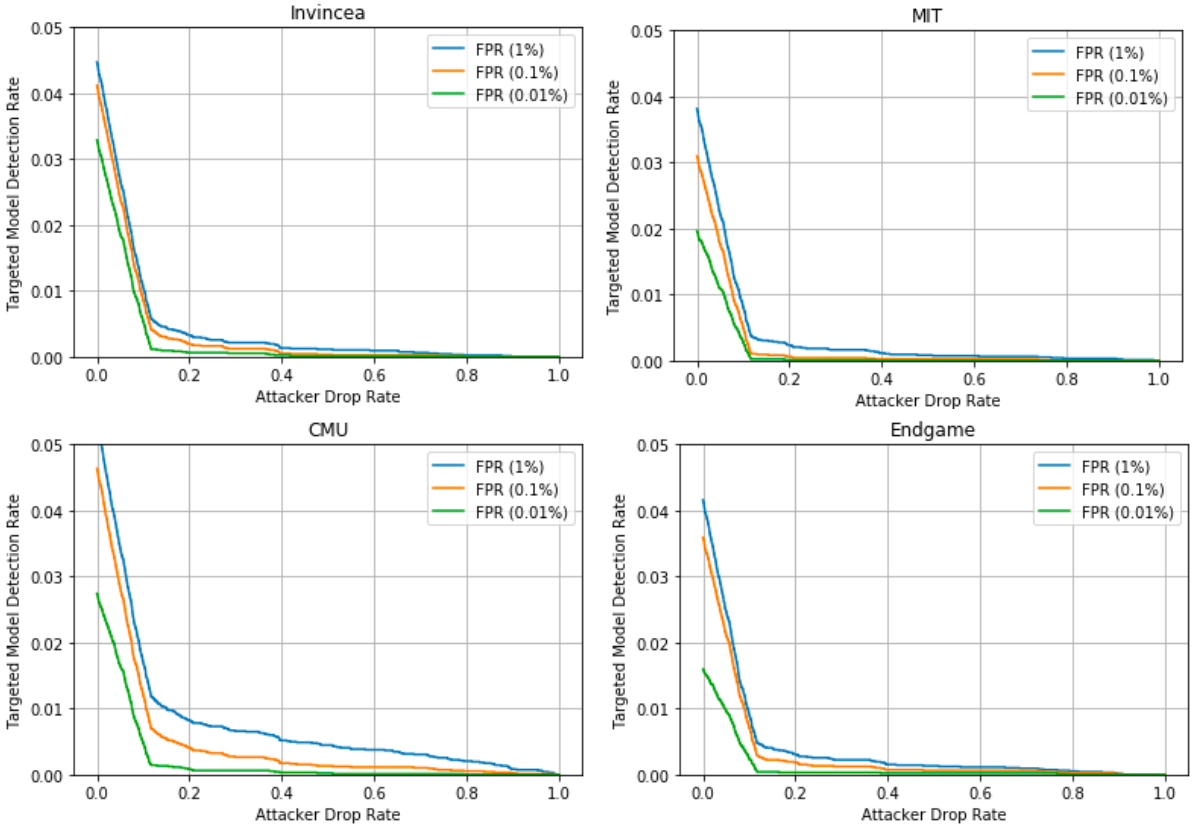


FIGURE 4. The rate of adversarial domain names that were successfully detected per drop rate configuration by every target model.

substitute model's score. The discarding of AGD names starts from the lowest to the highest substitute model score until the discard rate matches the drop rate.

We evaluate the true positive rate of targeted DGA classifiers based against an adversary that uses varying drop rate values of the MaskDGA attack. For every targeted DGA classifier the test includes three different settings of accepted false positive rates (FPRs): 1%, 0.1%, and 0.01%, which are commonly used for the analysis of DGA detection systems (see [25]). Our results indicate that increasing the drop rate significantly degrades the detection rate and inversely improves the evasion rate (as displayed in Figure 4). For example, if an adversary targets a CMU DGA classifier that was tuned for an acceptable false positive rate of 0.1%, then initially 4.5% of all of the adversarial domain names would be detected (see Figure 4). However, if the adversary sets the drop rate value at 0.1, then 10% of the generated adversarial domain names that were assigned with the lowest score by the substitute model would be discarded by both the adversary and its controlled bots immediately after generation, which results in the detection of only 1% of the adversarial domains.

VII. RELATED WORK

The concept of evading DGA classifiers was proposed by Anderson *et al.* that introduced DeepDGA [3] technique, Peck *et al.* that proposed the CharBot technique [25], and Spooren *et al.* that proposed the DeceptionDGA technique [29]. The DeepDGA technique utilizes a generative adversarial learning model based on the character distribution of benign domain names. The CharBot technique also deals with generateing domain names that will be classified as benign, by sampling legitimate domain names and arbitrarily replacing two characters with uniformly selected ones, while DeceptionDGA [29] attempts to evade detection by modifying values of linguistic features that are commonly used by machine learning models. DeceptionDGA was recently shown to be outperformed by CharBot [25], and accordingly, it was not included in our evaluation.

In contrast to these previous studies, MaskDGA is an evasion technique based on *adversarial learning*. The use of adversarial learning leverages known vulnerabilities of gradient-based classifiers such as inline DGA classifiers whose architecture relies on deep neural networks to degrade the accuracy of DGA classifiers even further than previously proposed studies of evasion techniques. Furthermore, MaskDGA can be used as a modular extension to DGAs that are already used by existing botnets, thus making it practical for adversaries. Additionally, this study is also the first to evaluate multiple adversarial attacks and adversarial

defenses using four state-of-the-art DGA classifiers in order to demonstrate the robustness and drawbacks of both the attacks and defenses.

MaskDGA can also be compared to the family of adversarial learning techniques against text classification models, similarly to TextBugger [17] and HotFlip [11]. MaskDGA differs from these techniques by relaxing the strict limitation of changing only a single character which is designed to avoid human perception. Instead, since MaskDGA is designed to evade DGA classifiers and not humans, an adversary can modify up to half the characters of an AGD name so that it will likely remain unreadable and therefore unregistered thus allowing the adversary to use it for its botnet operation.

The MaskDGA evasion attack operates within the input space (i.e., valid domain names) in contrast to studies of adversarial text attacks that operate directly in the embedding space. Gong *et al.* [12] proposed a translation of adversarial examples from the embedding space to the input space based on a nearest neighbor search. Their proposal guarantees that (1) embeddings are always translated to valid words, and (2) exchanged words preserve similar semantics based on the word mover's distance (WMD) metric. The translation to valid words using nearest neighbor search is clearly achievable for a corpus of a million English words, but without adaptation, it is infeasible for a corpus of domain names that are valid by the RFC1035 specification [19] which is 90 orders of magnitude larger. Additionally, the WMD metric guarantees that the input and the adversarial output preserve similar semantics, however, the notion of semantics is not well defined for algorithmically generated domains.

VIII. DISCUSSION

The analysis in Section IV-C argues that the requirements to train and operate MaskDGA are reasonable, even for machines with low computational resources. Nevertheless, considering the computational complexity and desired rate of evasion trade-off, we believe that given the attack use case, choice of evasion techniques other than MaskDGA are in place. For example, an adversary that operates in a well-protected environment, or that uses DGA for highly sophisticated malware (i.e., an APT) might prefer MaskDGA to assure evasion despite of the small computational overhead. In other cases, the adversary may prefer evasion techniques such as Charbot, which is easier to implement and is slightly less computationally demanding than MaskDGA.

IX. CONCLUSION AND FUTURE WORK

Inline character-level DGA classifiers have gained popularity in recent years due to their simplicity, the availability of training data, and the ability to use them for real-time detection. This study presents an adversarial machine learning evasion technique that exploits a vulnerability of inline DGA classifiers to evade detection.

The evaluation of MaskDGA on four state-of-the-art inline DGA classifiers shows that their detection rates decrease significantly in the face of MaskDGA and that they perform far worse in this situation than when existing evasion techniques (DeepDGA and CharBot) are applied, thereby demonstrating the clear advantage of adversarial learning for evading DGA classification. Furthermore, the use of adversarial learning in MaskDGA provides adversaries with a probability estimation for each domain generated in order to avoid detection, thus allowing adversaries to discard a portion of their generated domains for a better rate of detection by DGA classifiers, thus making MaskDGA even more powerful.

Additionally, an evaluation of the MaskDGA, DeepDGA, and CharBot evasion techniques against standard adversarial defenses, namely distillation networks and adversarial retraining, demonstrates that distillation is ineffective and that adversarial retraining is only effective against the specific adversarial samples on which it was trained and generally is ineffective against new adversarial samples (those that did not appear in the training set). These results demonstrate the vulnerability of character-level DGA classifiers to adversarial attacks.

Future work is encouraged on the topic of adversarial defenses in general, and in particular for inline DGA classifiers which are susceptible to existing evasion techniques. The DGA classifiers on which MaskDGA was evaluated were trained using supervised learning, which is the widely adopted approach to DGA classification due to the availability of datasets and high accuracy. However, it is possible that different approaches, which are less studied, might provide a more robust solution against MaskDGA, and adversarial attacks on DGA classifiers in general. Specifically, anomaly detection algorithms can be used to identify domain names that do not conform with existing benign or AGD names, and may thus be adversarial domain names. Another approach to be evaluated is combining classifiers using ensemble learning which is often considered more robust, despite still being susceptible to adversarial attacks [40].

Additional future work is encouraged to extend MaskDGA so it can operate directly in the embedding space. The main benefit of this is that adversaries that adopt DGA classifier architecture for their substitute models would be able to keep the non-differential embedding layer intact. This work can be carried out by leveraging the study of Gong et al [12] to search for adversarial examples within the embedding space.

X. MaskDGA DEMO CODE

The attack phase of MaskDGA is demonstrated using Python code in Listings 1.

GLOSSARY

AGD	Algorithmically generated domain
APT	Advanced persistent threat
DGA	Domain generation algorithm
DMD	Detection of malicious domains, a publicly available dataset of AGD names

```python
import numpy as np

def maskDGA(X, S):
    """
    X - One-hot encoded domain name
    S - Jacobian saliency matrix of X
    :return: Adversarial domain name
    """

    scores = np.trim_zeros(np.max(S, axis=1))
    median_score = np.percentile(scores, 50)

    adversarial = ""
    for pos_ohe, pos_gradients in zip(X, S):
        original_symbol = np.argmax(pos_ohe))
        new_symbol = np.argmax(pos_gradients)

        if np.max(pos_gradients) > median_score:
            adversarial += str(new_symbol)
        else:
            adversarial += str(original_symbol)

    return adversarial

#Example
X = np.array([[0, 1, 0],
              [1, 0, 0],
              [0, 0, 1],
              [0, 1, 0],
              [0, 0, 1],
              [0, 1, 0]])

S = np.array([[0.384, 0.058, 0.821],
            [0.155, 0.126, 0.521],
            [0.002, 0.674, 0.971],
            [0.125, 0.527, 0.189],
            [0.662, 0.770, 0.326],
            [0.574, 0.366, 0.282]])

print(maskDGA(X, S))
```

Listing. 1. MaskDGA attack phase Python code.

NOMENCLATURE

W	An input word (i.e., a domain name)
V	The set of allowed symbols in a domain name
$X_{\|W\|,\|V\|}$	A one-hot encoding representation of an input domain name (for brevity, often referred as X)
$F(X)$	The classification result of an input domain name. Ranging between zero (benign) and one (AGD).
Y^*	A binary label for a domain name. Can be either zero (benign) or one (AGD)
$\mathcal{L}(F(X), Y^*)$	The target loss
$S(X)$	The saliency matrix for X
$S(X)_{i,j}$	The saliency value of a feature $X_{i,j}$

ACKNOWLEDGMENT

This research was partially supported by the CONCORDIA project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 830927.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16, 2016, pp. 265–283.

[2] E. Alomari, S. Manickam, B. B. Gupta, S. Karuppayah, and R. Alfaris, "Botnet-based distributed denial of service (DDoS) attacks on Web servers: Classification and art," 2012, *arXiv:1208.0403*. [Online]. Available: http://arxiv.org/abs/1208.0403

[3] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur. (ALSec)*, 2016, pp. 13–21.

[4] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of DGA domain names," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*. IEEE, 2018, pp. 1–8.

[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[6] C. Catania, S. García, and P. Torres, "An analysis of convolutional neural networks for detecting DGA," in *Proc. 25th Congreso Argentino de Ciencias de la Computación (La Plata*, 2018, pp. 1–10.

[7] T. Chin, K. Xiong, C. Hu, and Y. Li, "A machine learning framework for studying domain generation algorithm (DGA)-based malware," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Cham, Switzerland: Springer, 2018.

[8] C. Choudhary, R. Sivaguru, M. Pereira, B. Yu, A. C. Nascimento, and M. De Cock, "Algorithmically generated domain detection and malware family classification," in *Proc. Int. Symp. Secur. Comput. Commun.* Singapore: Springer, 2018.

[9] A. Collette *et al.* (Mar. 18, 2017). *H5Py/H5Py:2.4.0*. Software. [Online]. Available: https://doi.org/10.5281/zenodo.400660

[10] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen, "Tweet2 Vec: Character-based distributed representations for social media," 2016, *arXiv:1605.03481*. [Online]. Available: http://arxiv.org/abs/1605.03481

[11] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," 2017, *arXiv:1712.06751*. [Online]. Available: https://arxiv.org/abs/1712.06751

[12] Z. Gong, W. Wang, B. Li, D. Song, and W.-S. Ku, "Adversarial texts with gradient methods," 2018, *arXiv:1801.07175*. [Online]. Available: http://arxiv.org/abs/1801.07175

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: https://arxiv.org/abs/1412.6572

[14] J. Huang, P. Wang, T. Zang, Q. Qiang, Y. Wang, and M. Yu, "Detecting domain generation algorithms with convolutional neural language models," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Communications/ 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 1360–1367.

[15] S. Joshua and K. Berlin, "eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys," 2017, *arXiv:1702.08568*. [Online]. Available: https://arxiv.org/abs/1702.08568

[16] J. J. Koh and B. Rhodes, "Inline detection of domain generation algorithms with context-sensitive word embeddings," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2966–2971.

[17] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "TextBugger: Generating adversarial text against real-world applications," 2018, *arXiv:1812.05271*. [Online]. Available: http://arxiv.org/abs/1812.05271

[18] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, "DGA botnet detection using supervised learning methods," in *Proc. 8th Int. Symp. Inf. Commun. Technol. (SoICT)*, 2017, pp. 211–218, doi: 10.1145/3155133.3155166.

[19] P. Mockapetris, *Domain Names-Implementation and Specification*, document RFC-1035, Network Working Group 55, 1987.

[20] A. Nadler, A. Aminov, and A. Shabtai, "Detection of malicious and low throughput data exfiltration over the DNS protocol," *Comput. Secur.*, vol. 80, pp. 36–53, Jan. 2019.

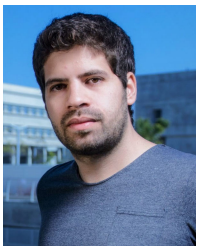
[21] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *Proc. MILCOM-IEEE Mil. Commun. Conf.*, Nov. 2016, pp. 49–54, doi: 10.1109/MILCOM.2016.7795300.

[22] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[23] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.

[24] A. Pathak, F. Qian, Y. C. Hu, Z. M. Mao, and S. Ranjan, "Botnet spam campaigns can be long lasting: Evidence, implications, and analysis," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 13–24, 2009.

[25] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, A. Nascimento, and M. De Cock, "CharBot: A simple and effective method for evading DGA classifiers," *IEEE Access*, vol. 7, pp. 91759–91771, 2019.
[26] D. Plohmann, "DGArchive," Tech. Rep., 2015.
[27] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 263–278.
[28] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. D. Cock, "An evaluation of DGA classifiers," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5058–5067.
[29] 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 *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2019, pp. 1902–1910.
[30] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, Jan. 2018, doi: 10.1016/j.neucom.2017.11.018.
[31] R. Vinayakumar, P. Poornachandran, and K. P. Soman, "Scalable framework for cyber threat situational awareness based on domain name systems data analysis," in *Big Data in Engineering Applications* Singapore: Springer, 2018, pp. 113–142.
[32] S. Vosoughi, P. Vijayaraghavan, and D. Roy, "Tweet2 Vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2016, pp. 1041–1044.
[33] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, *arXiv:1611.00791*. [Online]. Available: http://arxiv.org/abs/1611.00791
[34] B. Yu, D. L. Gray, J. Pan, M. D. Cock, and A. C. A. Nascimento, "Inline DGA detection with deep networks," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 683–692.
[35] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of DGA domain names," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*. IEEE, 2018, pp. 1–8.
[36] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
[37] F. Zeng, S. Chang, and X. Wan, "Classification for DGA-based malicious domain names with deep learning architectures," in *Proc. 2nd Int. Conf. Appl. Math. Inf. Technol.*, 2017, p. 5.
[38] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through DNS data analysis," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–36, Sep. 2018.
[39] S. Zhou, L. Lin, J. Yuan, F. Wang, Z. Ling, and J. Cui, "CNN-based DGA detection with high coverage," in *Proc. IEEE Int. Conf. Intell. Secur. Inform. (ISI)*, Jul. 2019, pp. 62–67.
[40] F. Zhang, Y. Wang, S. Liu, and H. Wang, "Decision-based evasion attacks on tree ensemble classifiers," in *World Wide Web*. Springer, 2020, pp. 1–21.



LIOR SIDI received the B.Sc. and M.Sc. degrees from the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, where he examined the application of deep learning, embedding, and adversarial learning in the fraud and cyber-security domains. He is currently a Data Science and a Machine Learning Researcher.



ASAF NADLER is currently pursuing the Ph.D. degree with the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev.

His primary research interest is the extraction of threat intelligence from large-scale DNS traffic using machine learning.



ASAF SHABTAI is currently a Professor with the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev. His main research interests are computer and network security, machine learning, the security of IoT and smart mobile devices, and the security of avionic and operational technology (OT) systems.