I. J. Computer Network and Information Security, **2023, 2, 1-14**
Published Online on April 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2023.02.01

Evaluation of GAN-based Models for Phishing URL Classifiers

Thi Thanh Thuy Pham
Academy of People Security, 125 Tran Phu Street, Ha Dong District, 12100, Ha Noi, Vietnam
E-mail: thanh-thuy.pham@mica.edu.vn
ORCID iD: https://orcid.org/0000-0003-3985-3599

Tuan Dung Pham
VNU University of Engineering and Technology, E3 Building, 144 Xuan Thuy Street, Cau Giay District, 11310, Ha Noi, Vietnam
E-mail: dungpt98@vnu.edu.vn
ORCID iD: https://orcid.org/0000-0002-2183-4640

Viet Cuong Ta*
VNU University of Engineering and Technology, E3 Building, 144 Xuan Thuy Street, Cau Giay District, 11310, Ha Noi, Vietnam
E-mail: cuongtv@vnu.edu.vn
ORCID iD: https://orcid.org/0000-0001-8058-5915
*Corresponding Author

Received: 20 October 2022; Revised: 01 November 2022; Accepted: 01 December 2022; Published: 08 April 2023

Abstract: Phishing attacks by malicious URL/web links are common nowadays. The user data, such as login credentials and credit card numbers can be stolen by their careless clicking on these links. Moreover, this can lead to installation of malware on the target systems to freeze their activities, perform ransomware attack or reveal sensitive information. Recently, GAN-based models have been attractive for anti-phishing URLs. The general motivation is using Generator network (G) to generate fake URL strings and Discriminator network (D) to distinguish the real and the fake URL samples. This is operated in adversarial way between G and D so that the synthesized URL samples by G become more and more similar to the real ones. From the perspective of cybersecurity defense, GAN-based motivation can be exploited for D as a phishing URL detector or classifier. This means after training GAN on both malign and benign URL strings, a strong classifier/detector D can be achieved. From the perspective of cyberattack, the attackers would like to to create fake URLs that are as close to the real ones as possible to perform phishing attacks. This makes them easier to fool users and detectors. In the related proposals, GAN-based models are mainly exploited for anti-phishing URLs. There have been no evaluations specific for GAN-generated fake URLs. The attacker can make use of these URL strings for phishing attacks. In this work, we propose to use TLD (Top-level Domain) and SSIM (Structural Similarity Index Score) scores for evaluation the GAN-synthesized URL strings in terms of the structural similariy with the real ones. The more similar in the structure of the GAN-generated URLs are to the real ones, the more likely they are to fool the classifiers. Different GAN models from basic GAN to others GAN extensions of DCGAN, WGAN, SEQGAN are explored in this work. We show from the intensive experiments that D classifier of basic GAN and DCGAN surpasses other GAN models of WGAN and SegGAN. The effectiveness of the fake URL patterns generated from SeqGAN is the best compared to other GAN models in both structural similarity and the ability in deceiving the phishing URL classifiers of LSTM (Long Short Term Memory) and RF (Random Forest).

Index Terms: Attacker, Defender, Discriminator Network, Fake URLs, Phishing URL Detection.

1. Introduction

A phishing URL is a clickable link that directs network users to a malicious or otherwise fraudulent web pages. Phishing attack by malicious URL/web links is one type of social engineering attacks. The attacker tries to lure the users into clicking on these links for negative intents, such as identity theft, malware infection to the target systems. The scammers masquer-ade trusted entities or well-known companies to deliver malicious URLs to the victims' systems

through emails, messages, websites, or advertisements. Of these, emails is the most widely used method to distribute phishing URLs. According to Symantec's 2019 Internet Security Threat Report (ISTR) [1], the top five subject lines for business email compromise (BEC) attack are Urgent, Request, Important, Payment, Attention. Research from Cofense in 2021 [2] suggests phishing emails are slightly more like to contain a link to a malicious website (38%) than a malicious attachment (36%). In 2021 Terranova's report [3], almost 1 in 5 of all employees are likely to click on phishing links sent via emails. Due to globally-risen mobile use, SMS (Short Message Service)/Text phishing or Smishing attacks have risen dramatically in recent years. According to data from enterprise security provider Proofpoint, this type of attack increased by almost 700% in the first six months of 2021. A common way of Smishing attack is to include malicious short links in the text messages sent to the victim's smartphones.

Several anti-phishing URLs solutions have been proposed. Generally, they can be categorized into two approaches: (1) using blacklist or whitelist for phishing URL filtering and (2) using machine learning methods for training URL classifiers or URL phishing detectors. The first approach is often used by search engines to give warnings to users for the malicious URLs matched with the ones in blacklist. However, the blacklist is not effective enough to block new links. In reality, most malicious URLs are newly created and not promptly updated in the blacklist. The second approach is increasingly attractive for research and application development. Traditional machine learning algorithms such as SVM, Naive Bayes, Random Forest etc. have recently been replaced by deep learning models to train URL classifiers. Based on large and diverse database of URL samples, the efficient URL classifiers or detectors can be built [4,5,6].

In this work, we explore different GAN models, from the basic model to others GAN extensions of DCGAN, WGAN, SEQGAN for two investigations: (1) the success of the Discriminator networks in different GAN models for phishing URL classification. This helps to give a suggestion to select a robust GAN model for anti-phishing URL; (2) The effectiveness of the GAN-generated URL strings in terms of deceiving the phishing URL classifiers of LSTM and RF. The more similar in the structure of the GAN-generated URLs are to the real ones, the more likely they are to fool the classifiers. This investigation is done by (a) applying two evaluation scores of TLD and SSIM for measuring the structural similarity between the fake URLs generated by GANs and the real ones; (b) training the classifiers of LSTM and RF in two scenarios of training dataset: (i) training on the original balanced datase (OBD which contains malign and benign URL samples and (ii) OBD added with the GAN-generated URLs (OBD+GANgen). This shows the feasibility of fighting URL classifiers/defenders by the URLs generated by GAN models. And from this, more specific researches and efforts are made to defend against these potential opponents.

2. Related Works

The GAN architecture is first proposed for generating images by [7]. It includes two neural networks contesting with each other in zero-sum game. The Generator synthesizes samples that appear to be from a desired distribution. The Discriminator is responsible for distinguishing between synthetic samples and true population. Variations of GANs can be split into several categories. For examples, DCGAN [8] and LAP-GAN [9] replace the model architecture of Generator and Discriminator for better approximating the data distribution. The works of WGAN [10] and WGAN-GP [11] alter the loss function of GAN to stablize the training of GANs. In application domains, apart from generating image output from the noise space, GAN-based architecture can be used on a wide range of tasks. CycleGAN [12] employs a cycle consistent loss to map from one domain to other domains without the need the pairing source and target images. For text generation purpose, the works of [13-15] introduce the adversarial training into generating sentences. For example, SeqGAN [13] uses a reward-based training to finetune the decision in selecting the next token for constructing a sequence.

In anti-phishing URL, early proposals exploit rule-based framework [16-18] for URL sample generation and traditional machine learning classifiers, such as SVM, Random Forest, etc for URL phishing detection. Recently, GAN models are proposed to generate the synthetic adversarial samples and enhance the detection ability of the classifiers by training the classifiers with the generated ones. The synthesized samples are at character or feature levels. They are exploited from a specific part of URL string (mostly domain name part) and the whole URL string.

In [19] a GAN model is proposed to pseudo-randomly generate domain names on a character-by-character basis. Legal domain names are used in training phase with an autoencoder architecture. The Discriminator in the proposed GAN model is the encoder network which is responsibility for domain embedding. The decoder network plays the role of Generator which gets input of domain embedding and outputs a domain. A random forest classifier is used to detect the stealth ability of the domain names that are synthesized from the proposed GAN model. Although the generated domains by GAN could be used for training phase of anti-phishing URL classifiers to improve their learning capability, the GAN training is unstable. Several GAN training techniques should be considered [20-21]. The authors in [22] proposed three variants of GANs for domain name generation that have similar characteristics to the available benign domains for fraudulence. It is similar to [19], the auto-encoder architecture is applied for the GAN variants of Least Squares GAN (LSGAN), Wasserstein GAN with Gradient Penalty (WGAN-GP), and the original GAN. However, it differs from [19] in that it not only evaluates the effectiveness of fooling the classifiers, but also extends to other evaluation criteria of domain length, domain collision, repeated domain collision, unigram and bigram distributions of generated samples in comparison with the original ones. In [22], the classifiers of Endgame, Invincea, CMU, MIT, NYU [23] are implemented to detect the adversarial domains. The experimental results show that WGAN-GP brings the best performance in all evaluation criteria compare to other GAN variants.

The work in [24] utilized a basic GAN structure which consists of a Generator network, a Discriminator network and a Blackbox Phishing Detector. The GAN networks in [19, 22] work on character level of domain name but in [24], the GAN model used binarized versions of intra-URL features, such as the existence of "HTTPs" token, a prefix or suffix separated by "-" in the domain, the number of words in the remaining part of an URL.The Generator is used to generate adversarial samples. The Discriminator learns to fit the phishing-URL detector, which is implemented using a specific classification algorithm to identify phishing samples. At each round of the training process, the Discriminator sends feedback to the Generator to modify its weights during the training process to the point where it guarantees that the Generator creates enough samples to evade the phishing URL detector. The experimental results demonstrated that GAN is quite effective for deceiving the classifiers that are created to defeat sophisticated attacking attempts of exploiting intra-URL relationships. The results also show that adversaries may focus on highly ranked features and can still bypass machine learning-based phishing URL detection techniques. The most important implication of this work is to create countermeasures to mitigate adversarial samples.

A variant of GAN network named WGAN-GP is prpoposed in [11] for generating character-level malicious URL strings from the available phishing URL data. This solves the shortage problem of malign URL strings for training deep learning models. From the input training set of benign and malign URLs, a GAN model is trained to differ between the input malicious URLs and the new URLs generated from the Generator network. The classifiers of LSTM and GRU are trained on the extended dataset to show the improvement of classification accuracy when adding generated malicious URL strings to the original training set. Another GAN variant called Conditional GAN (CGAN) is proposed in [25] with character-level data of URL strings. It is different from other unconditional GAN-based approaches, the Generator network of CGAN gets the input of not only the random noise but also the original data (malicious or benign URLs). This hopes to give out the more realistic outputs. The Discriminator takes both clean URLs and phishing URLs as inputs sequentially. It simultaneously predicts if the generated samples are real or fake and classifies them as benign or malign ones.

In this work, the basic GAN model and other GAN variants of DCGAN, WGAN and SeqGAN are explored for evaluating the role of the Discriminator networks in different GAN models for phishing URL detection. We train the Discriminator network in turn on two sets of benign and malign URL strings. Based on this, the Discriminator can learn to discriminate two real distributions of clean and phishing samples in the testing set. We prove that basic GAN and DCGAN gain the best classification performance by their Discriminator networks. In addition, it is different from other GAN-based proposals, in this work we train GAN models on the original dataset of clean and phishing URLs to generate the new URL samples. These newly generated URL samples will be added to the train set with aim at data augmentation for training malicious URL detectors. The adversarial URL samples synthesized by different GAN models are evaluated by 2 criteria: (1) the structural similarity of the artificial URL strings with the real ones; (2) the ability of fooling classifiers trained on the dataset with the addition of artificial URL samples compared to the dataset without this addition. These evaluation criteria derive from the fact that the attackers would like to create malicious URL samples as close to the real ones as possible to deceive users and classifiers.

3. Proposed Methods

In this work, two GAN-based investigations are setup for evaluations: (1) the performance of the Discriminator networks in phishing URL classification; (2) the performance of the adversarial URL strings synthesized from the Generator networks. The basic GAN model and three GAN variants of DCGAN [8], WGAN [10] and SeqGAN [13] are exploited for each investigation.

3.1. Overall Frameworks

The overall framework for the first investigation is shown in Fig 1. It includes two phases of training and testing. In training phase, we consider both the malign URL strings and Benign URL strings which collects from real world as the real URL strings. The Generator in GAN is used to create fake URL. The task of Discriminator is to classify between the real and the synthesized URL strings. This helps to enhance the ability to learn the clean and phishing URL distributions of the Discriminator network. The trained Discriminator network is then used for URL classification in the testing phase. The testing set consists of clean and phishing URL strings. The classification accuracy of the Discriminator network is evaluated through each round of training and testing on each type of benign or malign URL strings. A threshold is set for this evaluation based on calculating the average score obtained with clean and phishing data

The framework for second investigation is shown in Fig 2. It also contains two phases. The first phase is also to train a Discriminator and Generator of GAN. In this scenario, we are focus on the capacity of Generator to generate synthesized URL strings. We then measure the quality of the generated fake synthesized URL strings from the Generator by using the synthesized URL strings to train several baseline classifiers, which are the LSTM and RF. More specifically, two scenarios are set: (1) the baseline classifier is trained with the real sets of malign and benign URL strings; (2) the baseline classifier is trained with a real set of benign URL strings and a synthetic set of URL strings. The synthesized URL set is produced by training GAN model with the real set of both malign and benign URL strings. This GAN training aims at generating the fake URLs which are as close to the distributions of the real URLs as possible. After the phase of classifier training, the testing phase is done. A test set of the clean and phishing URLs is used for evaluating the classification

performance from the two training scenarios. This comparative evaluation is intended to detect GANs' ability in generating adversarial URL strings from the real set of URLs. This is helpful in knowing the attack behavior can be done by GANs as well as the defending direction by training the classifiers with the augmented set of GAN-based generated phishing URLs.

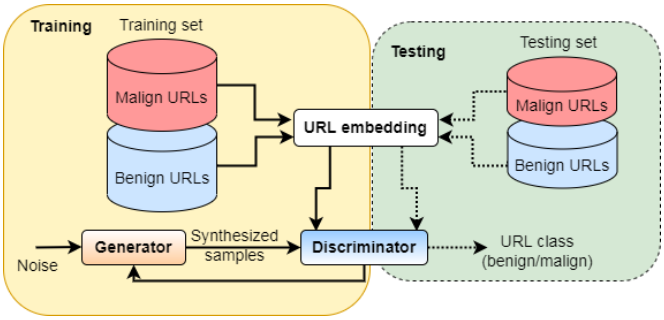


Fig.1. GAN training and testing phases for phishing URL classification by the Discriminator network.

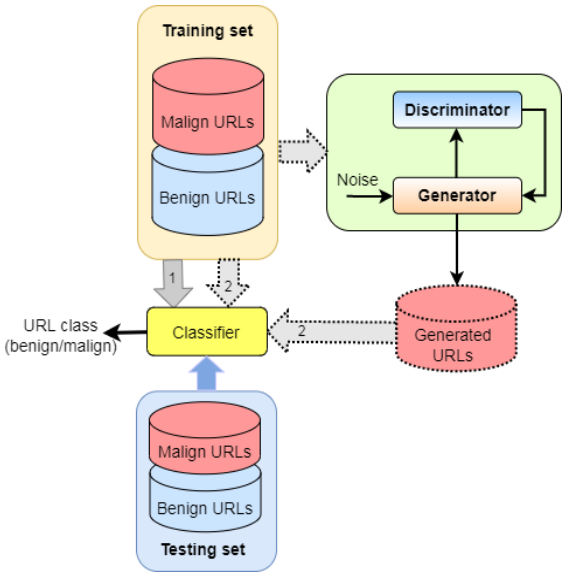


Fig.2. The training and testing phases of the URL classifiers.

3.2. Experimented GAN Models

A. Basic GAN

The main idea of GAN is to train two adversarial networks of Discriminator and Generator. The task of Generator **G** is to generate the samples as close as possible to the target distributions. The goal of Discriminator **D** is to distinguish real and fake samples generated by **G** network.

Let p_z is the probability distribution of the noise input z from a pre-defined distribution, p_g is the distribution of the Generator **G** over the real URL samples and p_r is the distribution of x that need to be generated.

In GAN-based URL generation, a real URL sample x could be a clean or phishing sample. Although it is a trivial task to generate an URL-like sequence of characters, the main idea of using the GAN framework is to examine the effects of the distribution distance between p_g and p_z for two downstream tasks in Fig. 1 and Fig. 2.

In the basic GAN approach, the distribution distance is indicated by the Discriminator **D**. Given an input x, the Discriminator **D** would try to predict the probability $D(x)$ which indicates whether x is a real or a generated one. Mean while, **G** tries to generate sample x from an input noise z. Both **D** and **G** are parameterized by a deep network and trained by minimizing the loss function:

$$\begin{aligned}\min_G \max_D L(D,G) &= \mathbb{E}_x[logD(x)] + \mathbb{E}_z\left[\log\left(1 - D\left(G(z)\right)\right)\right] \\ &= \mathbb{E}_x[logD(x)] + \mathbb{E}_x\left[\log\left(1 - logD(x)\right)\right]\end{aligned} \quad (1)$$

The main contributions of GAN to generation model are two folds. Firstly, GAN switches the traditional loss function of generative models from the KL (Kullback-Leibler) asymmetric distance to the symmetric JS (Jensen-Shannon Divergence). It is well-known that training GAN (specifically training **D**) by KL distance is hard to be calculated.

Meanwhile, the JS distance can be approximated through **D**. Secondly, the GAN training setup allows both **G** and **D** compete together to improve each other performance from the initial configures.

The advantage of this approach is to prevent the phenomenon of model collapse when the JS distance is improved too fast compare to the capability of the generative model.

The main contributions of GAN to generation model are two folds. Firstly, GAN switches the traditional loss function of generative models from the KL (Kullback-Leibler) asymmetric distance to the symmetric JS (Jensen-Shannon Divergence). It is well-known that training GAN (specifically training **D**) by KL distance is hard to be calculated. Meanwhile, the JS distance can be approximated through D. Secondly, the GAN training setup allows both **G** and **D** compete together to improve each other performance from the initial configures. The advantage of this approach is to prevent the phenomenon of model collapse when the JS distance is improved too fast compare to the capability of the generative model.

Although the original proposal of GAN is to generate images, there are several ways to parameterize **D** and **G** to train on the other data structures. In the domain of URL strings, a GAN-based generated URL is a 1-D data structure with each element is a character. Although an URL string can be seen as textual form, the contextual structure of the URL is more restricted. In our work, we vary the **G** and **D** models and the loss function in the Eq. 1 to work with URLs generation task.

The most straightforward approach is to utilize the standard feed forward layer. In our Basic GAN architecture, we stack the feed forward layer in **G** and **D** to generate the desired outputs. The **G** network generates a URL sequence from the input noise z with several feed forward layers. Each layer works as a transform operator which transforms the intermediate representation to a higher dimension. The last layer would connect to the output layer, which represents an URL string. The Discriminator **D** works in the reversed way. It transforms the URL from the URL space to the output probability.

B. DCGAN

In the DCGAN architecture [8], the feed forward layer is replaced with the convolution and de-convolution layers. The DCGAN structure is extended directly from the basic GAN. It is firstly designed to work with image generation tasks. The layers in DCGAN leverage the standard convolution architecture from the traditional supervised learning tasks on image into the unsupervised learning tasks. In the Discriminator structure of image generation task, the implementation of DCGAN layers alternates between convolution layers, batch normalization and non-linear activation functions. In the Generator structure, the input vector represents a 2D noise vector. It is then transformed by de-convolution layers into a 3-dimension volume which represents the image dimension.

Based on the standard DCGAN structure, our work extends the model for generating URL strings. We can view an URL string as a 1-D image in which each value is selected from a list of available values. The feed forward layers in the Basic GAN model are replaced by the convolution 1D and de-convolution 1D in the DCGAN. From the learning perspective, the convolution and de-convolution layers add a constraint between consecutive characters. The number of consecutive characters is characterized by the size of receptive field.

C. WGAN

In the process of GAN training, the Discriminator **D** should provides a smooth signal to update the Generator **G**. Normally, it is hard to achieve this property with the standard KL loss or JS loss. One of the main reason is that in low manifold of data distribution, it easily distinguish real distribution from fake distribution. Therefore, the Discriminator **D** can quickly converge and hurts the training process of the Generator **G**. This phenomenon sometimes called the model collapse. In the task of URL generation, the model collapse happens when the Generator **G** fails to generate the proper URLs from the noise signal z.

In order to stabilize the training process, WGAN [10] employs the Wasserstein distance instead of standard JS divergence. According to the authors of WGAN, the Wasserstein distance results in a more smooth measure in the low dimension manifold than the KL or JS divergence. The training loss in \ref{eq:basicgan} is updated with the WGAN critic loss as follows:

$$L = \mathop{\mathbb{E}}_{\hat{v}\sim\mathbb{P}_g}[D(\hat{v})] - \mathop{\mathbb{E}}_{v\sim\mathbb{P}_r}[D(v)] \tag{2}$$

where $\mathbb{P}_r$ is the distribution of URL sample and $\mathbb{P}_g$ is the model distribution, defined by $\hat{v} = G(z), z \sim p(z)$. In detail, $\mathbb{P}_z$ represents the distribution of noise z (the distribution could be Gaussian, uniform, etc...). In order to enforce the K-Lipschiltz continuity, a weight clipping step is used for updating the weight of **D**.

Wasserstein GAN with Gradient Penalty

The WGAN-GP loss function is set based on the original critic loss of Wasserstein GAN (The Wasserstein distance) combined with a proportion of gradient penalty:

$$L = \underbrace{\mathop{\mathbb{E}}_{\hat{v}\sim\mathbb{P}_g}[D(\hat{v})] - \mathop{\mathbb{E}}_{v\sim\mathbb{P}_r}[D(v)]}_{\text{WGAN critic loss}} + \lambda \underbrace{\mathop{\mathbb{E}}_{\hat{v}\sim\mathbb{P}_{\hat{v}}}[(\|\nabla_{\hat{v}}D(\hat{v})\|_2 - 1)^2]}_{\text{Gradient Penalty}} \quad (3)$$

The gradient penalty in Formula (3) is the penalty on the gradient norm for random URL $\hat{v} \sim \mathbb{P}_{\hat{v}}$. The original WGAN uses weight clipping to enforce the Lipchitz constraint. However, this approach has disadvantages of exploding and vanishing gradients. The gradient penalty is an alternative method to make sure the Lipschitz constraint is satisfied. With an input noise $z \sim \mathbb{P}_z$, to learn the Generator distribution $\mathbb{P}_G$ over data $\hat{u}$, the Generator **G** will try to produce output $G(z)$. This output becomes the input for the Discriminator **D**. In other GAN variations, the **D** is used as a classifier and outputs a probability which implies whether the input is real or fake sample. Discriminator in WGAN-GP is not a classifier but a critic for scoring the real or fake of samples. From the Formula (3), we see that our objective is to maximize the score for real samples and minimize the score for fake samples.

The Generator learns to generate fake data by maximizing the feedback score from the Discriminator network $D(\hat{v})$. The input of the Generator is a noise z which is randomly sampled. All convolution layers in our work are 1-dimension convolutions since we only process textual connected layer followed by a number of residual blocks and a convolution layer. At the end of the Generator is a Softmax layer. The residual block follows the architecture of [26], which contains two Convolution layers, two ReLU activation layers and a residual connection. This connection skips the layers in the network and then combined with the output using α as a measurement of how much we want to combined with the output. The output $\hat{v}$ of the Generator is feed into the Discriminator for evaluation and this process is repeated for a number of iterations.

The Discriminator Network

The Discriminator tries to minimize the Wasserstein distance $D(\hat{v}) - D(v)$. The network tries to score the fake data $\hat{v}$ generated by the Generator network and the real data v which is the malign URL in our case. The quality of the data generated by the Generator is related to Wasserstein distance. The smaller the score between the fake and real data implies the better similarity between the real distribution and the fake distribution. The Discriminator architecture contains a Convolution layer, followed by a number of Res Blocks. After the Flatten and Fully Connected layers, the Discriminator outputs a critic score. During training the Discriminator network, the Generator network is not trained. The weights of the Generator network remains constant while it generates training samples for the Discriminator network.

D. SeqGAN

The SeqGAN is designed to work for sequence generation by modeling the generation process as a chain of actions. Each action chooses the next token based on the previous token. The objective of SeqGAN training is to maximize the expected reward as an reinforcement learning optimization task. In our URL generation task, the action of SeqGAN is to choose a character from a set of valid characters. An URL sequence is then generated by several actions until there is an ending character. The reward from the sequence of actions is then assigned by the Discriminator of GAN. The Generator G is a policy, which is parameterized by a deep network θ. It is more specifically, the generated URL is modeled by a sequence $y = (y_1, .., y_t, .., y_T)$, where $y_t \in \mathcal{Y}, \mathcal{Y}$ is a set of possible characters in an URL string and T denotes the number of actions. The set of possible characters is extracted from all URL strings in the training data. The number of actions T also denotes the length of generated actions. At each roll-out, the starting input state is set to the empty string. The Generator **G** receives the input state and chooses the next character by feeding the input state into the deep network θ. The output character is then appended into input state to reach a new state. The process is repeated until the ending character is generated. The length of generated URL is restricted by a maximum value.

We denote the input state s at time t as a sequence of the previous $t-1$ character, where y_1 is the first character of URL sequence: $y_{1:t-1} = (y_1, y_2, .., y_{t-1})$. The goal of SeqGAN model is to maximize the reward for completing the whole sequence given the initial character y_1. In order to achieve this, the objective function $\mathcal{J}(\theta)$must be maximized:

$$\mathcal{J}(\theta) = \mathbb{E}[(R_T|s_0, \theta)] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1|s_0) \cdot Q_{D_\emptyset}^{G_\theta}(s_0, y_1) \quad (4)$$

In the Equation (4), R_T is the reward for completing the URL sequence given the starting state s_0. The Generator G_θ is responsible for making the decision in choosing the next character. The Discriminator $D_\emptyset$ acts as a guide for the Generator by determining the probability that the generated URL sequence $Y_{1:T}$ is sampled from the real data distribution or not. Furthermore, this probability also used as the reward R_T for the Generator.

$$Q_{D_\emptyset}^{G_\theta}(s = Y_{1:T-1}, a = y_T) = D_\emptyset(Y_{1:T}) \quad (5)$$

Equation (5) expresses the expected accumulative reward starting from state s, taking action s with policy G_θ. Since the Discriminator only returns the reward for completing a full sequence, the intermediate reward is ignored. A model without intermediate reward is sparse and this makes it harder to learn than model with dense reward. In order to overcome this and calculate the intermediate reward, a long term reward will be taken into account. The future $T-t$ character

$(Y_{t+1:T} = (y_1, \ldots, y_T))$ with given the current state s_t can be sampled using Monte Carlo Search with the roll-out policy G_β. Follow [13], a $N - time$ Monte Carlo is defined as:

$$\{Y_{1:T}^1, .., Y_{1:T}^N\} = MC^{G_\beta}(Y_{1:T}; N) \tag{6}$$

The intermediate reward is then computed as follows:

$$Q_{D_\emptyset}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\emptyset(Y_{1:T}^n), & t < T \\ D_\emptyset(Y_{1:t}), & t = T \end{cases} \tag{7}$$

where $Y_{1:T}^n \in MC^{G_\theta}(Y_{1:T}; N)$. In the original SeqGAN implementation, the Generator is updated using Monte Carlo searching for N times. If N is too small, the model may not have enough exploration for a better reward.

4. Experiments and Results

4.1. Data Preparation

A. URL Embedding

In order to train and test GAN models, URL strings need to be encoded to numeric vectors. Given a set $U = (u_1, y_1), .., (u_i, y_i)$ with $(0 \leq i \leq n)$, where u_i is the ith URL string and y_i is the label of the URL, $y_i \in (0,1)$ indicates whether a URL is malign $(y_i = 1)$ or benign $(y_i = 0)$. Based on the length distribution of URL strings (Fig. 3), a length scale of m characters is set for each URL string to perform URL embedding. Thus, each URL u_i consists of m characters $(m = 200)$ represented by $u_i = u_i^1, .., u_i^m$. Each character in a URL string is encoded into a one-hot vector using a dictionary D of size d $(d = 129)$. The dictionary is created based on a statistic of the unique characters appear in the experimental dataset. They include ASCII and non-ASCII characters and symbols. Each unique character/symbol is assigned to an integer from 1 to 129 (see Fig. 4).

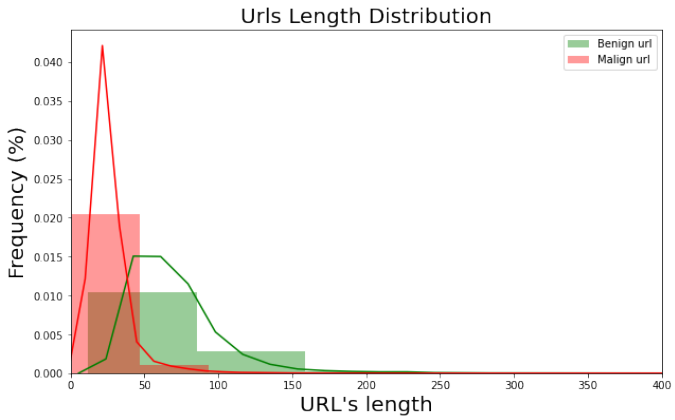


Fig.3. Length distribution of URL strings in the experimental dataset.

B. Experimental Dataset

Table 1. The experimental dataset with the number of training and testing URL samples.

Dataset	Training Set	Test Set
Phishing No	1M	1M
Clean No	1M	1M

The experimental dataset is constructed with 2 million legitimate URLs and 2 million phishing URLs[1], with the pre-processing step is done to eliminate any duplicate and invalid URL samples.

In order to evaluate ability of phishing URL classification by Discriminator networks in different GAN models (the first investigation), 1M of malign URLs in 2M dataset are used for Discriminator training. The phishing URL classification performance of trained Discriminator network is then evaluated on the test set of 1M phishing URLs and 1M benign URLs.

[1] https://data.mendeley.com/datasets/kvpkc4j658

```
{'t': 1, 'o': 2, 'e': 3, 'a': 4, '/': 5, 's': 6, '.': 7, 'i': 8, 'r': 9,
'n': 10, 'c': 11, 'l': 12, 'p': 13, '-': 14, 'm': 15, 'h': 16, 'd': 17,
'b': 18, 'g': 19, 'u': 20, 'w': 21, '0': 22, '1': 23, 'f': 24, '2': 25,
':': 26, 'y': 27, 'k': 28, '%': 29, 'v': 30, '8': 31, '3': 32, '5': 33,
'9': 34, '4': 35, '7': 36, '6': 37, 'x': 38, '=': 39, 'z': 40, '_': 41,
'j': 42, '&': 43, '?': 44, 'q': 45, '+': 46, ',': 47, ';': 48, '~': 49,
'(': 50, ')': 51, '@': 52, '!': 53, "'": 54, '\\': 55, '$': 56, '*': 57,
' ': 58, 'é': 59, '>': 60, '<': 61, '\u200a': 62, '|': 63, ']': 64, '[':
65, '}': 66, '{': 67, '#': 68, '–': 69, '`': 70, '’': 71, 'è': 72, 'ç': 7
3, 'ã': 74, '¿': 75, '^': 76, 'ó': 77, 'ï': 78, '½': 79, '©': 80, 'â': 8
1, '83 :',' ,82 :'›', 'š': 84, 'ô': 85, '“': 86, '”': 87, 'á': 88, 'ö': 89,
'à': 90, 'í': 91, '"': 92, '\xad': 93, '97 :'ב' ,96 :'ן' ,95 :'ו' ,94 :'ר',
'\x13': 98, 'û': 99, 'ü': 100, '°': 101, '´': 102, '€': 103, '¯': 104,
':'ט' ,111 :'צ' ,110 :'ק' ,109 :'ל' ,108 :'ש' ,107 :'ח' ,106 :'נ' ,105 :'ת
116 :'…' ,115 :'¼' ,114 :'כ' ,113 :'ח' ,112, 'ê': 117, '‰': 118, 'î': 119,
'—': 120, '®': 121, 'ž': 122, 'ä': 123, 'ƒ': 124, 'æ': 125, '§': 126,
'¢': 127, '成': 128, '品': 129}
```

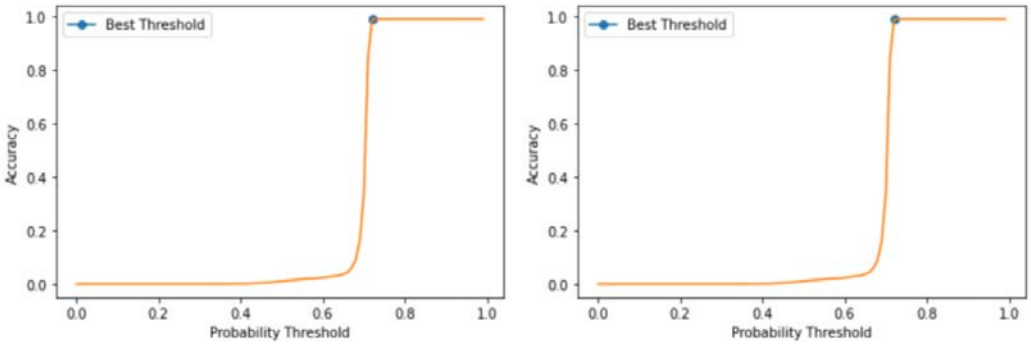
Fig.4. The dictionary for URL embedding.

In order to evaluate the effectiveness of URL strings synthesized from the Generator networks of different GAN models (the second investigation), 1M of benign and 1M of malign URLs are used for GAN training. The RF and LSTM classifiers are trained with two data scenarios: (1) using the same training dataset as the case of GAN training. It is called the original training dataset; (2) the training dataset contains both original set and the URL strings that are produced by GAN models. The number of the GAN-based generated URL strings are 1M for malign and 1M for benign ones.

4.2. Evaluation of Phishing URL Classification by Discriminator Networks

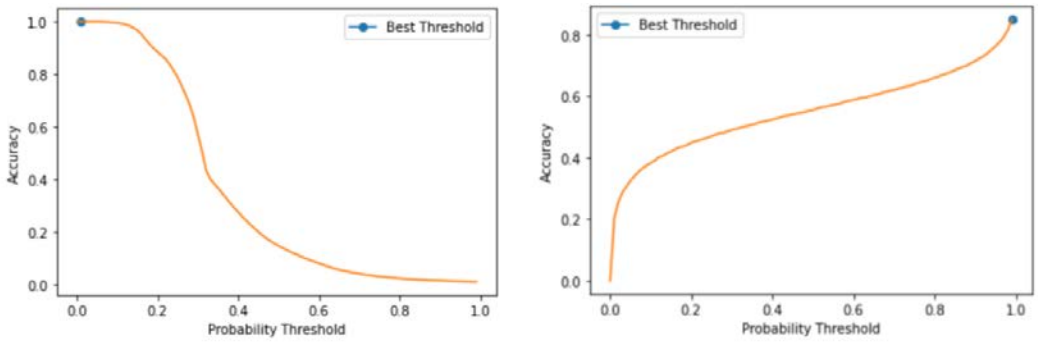
A. Threshold Searching Strategy

A threshold searching strategy is implemented to improve the classifiers' performance in predicting probabilities of the class labels on an imbalanced URL class distribution. The searching is started at a probability threshold of 0.00, then it is increased by 0.01 at each step. The accuracy of each GAN model is calculated at each step and the decision threshold relating to the best accuracy is chosen. The discrimination threshold values chosen for different models of GAN, DCGAN, WGAN and SeqGAN are shown in Fig. 5a, Fig. 5b and Fig. 6a, Fig. 6b, respectively.



(a) (b)

Fig.5. Threshold graphs and the chosen values (marked as a solid circle) for GAN model (a) and DCGAN model (b).



(a) (b)

Fig.6. Threshold graphs and the chosen values (marked as a solid circle) for WGAN model (a) and SeqGAN model (b).

With GAN, DCGAN and SeqGAN models, the chosen threshold values are 0.64, 0.72 and 1.00, respectively. These thresholds bring the best classification accuracy on the test set for the discriminator of each considered GAN model. GAN and DC GAN have quite similar architecture, with the only difference is that DC GAN utilizes convolutional layers instead of linear layers in GAN. This produces threshold values that do not differ much between GAN and DC GAN. SeqGAN discriminator is different since the input is constructed in a sequential manner, therefore the peak accuracy is reached when the threshold is set at 1.00. The best threshold is chosen at 0.00 for WGAN. This stems from the fact that the WGAN output is the Wasserstein distance, therefore only a certain range of the scores would shows the classification bound of the Discriminator. Thus, we choose the WGAN threshold value at the point where the classification accuracy is at the peak and then starting to decreased. Table 2 summarizes the classification thresholds chosen for each GAN model.

Table 2. The classification thresholds for different GAN models.

Models	Basic GAN	DCGAN	WGAN	SeqGAN
Thresholds	0.64	0.72	0.00	1.00

B. GAN Training Results

Basic GAN Model

In the basic GAN setup, we choose Adam as an optimizer with learning rate of 2e-5. The model is trained with 1,000 iterations in each phase. Figure 7 shows the Discriminator and Generator loss values through each iteration. Discriminator loss (d_loss) and Generator loss (g_loss) are approximately ranged from 0.75 to 1.00. It can be seen that both losses are stable at early running to iteration of 200, and slightly erratic after that.

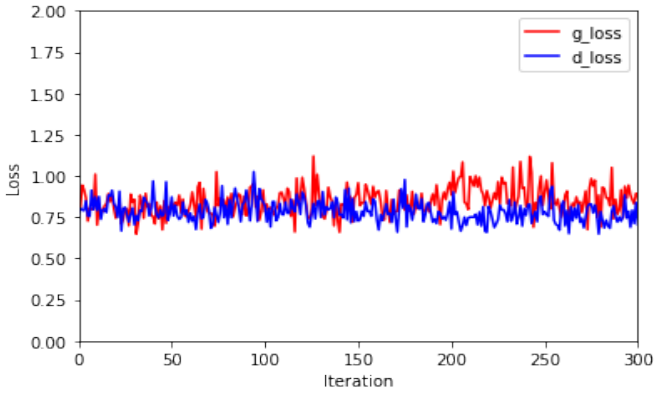


Fig.7. GAN loss graph over iterations.

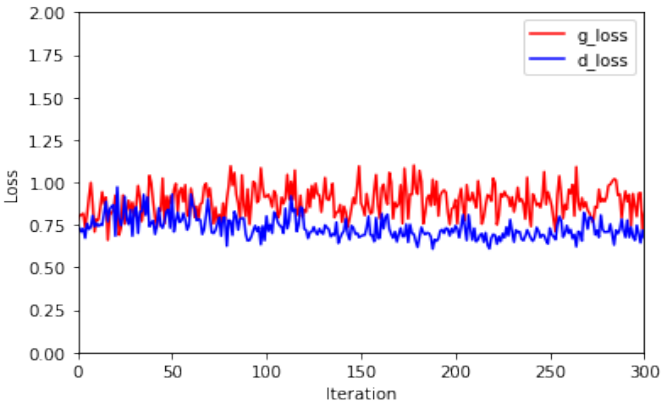


Fig.8. DC GAN loss graph over iterations.

DCGAN Model

In DCGAN setup, we choose Adam as an optimizer with learning rate of 2e-5, and train the model in 1,000 iterations for each phase. Fig. 18 shows the Discriminator and Generator loss values through each of 300 iterations. It can be seen that they are quite stable, around 0.75 to 1.00 for Discriminator network (d_loss) and from 0.75 to less than 1.25 for Generator network (g_loss). Both losses are quite stable at the first 70 iterations, then they move at the interval of 0.25 apart.

WGAN Model

The model is trained with a batch size of 64 for 3,000 iterations. The number of critics is 10 and the lambda hyperparameter for gradient penalty is 10. For the optimizer of the Generator and the Discriminator, Adam algorithm is chosen with learning rate of 1e-4, betas of 0.5 and 0.9 for each. Figure 9 shows the Discriminator and Generator loss values through each iteration. It can be seen that the Wasserstein distances of d_loss and g_loss are close to each other at the first few iterations and then become larger until 2000 iterations. Then the model starts to converge to a point near the iteration of 3000, and at which the gap between d_loss and g_loss is reduced and stable at around 0 to -100 for d_loss and near -100 for g_loss.

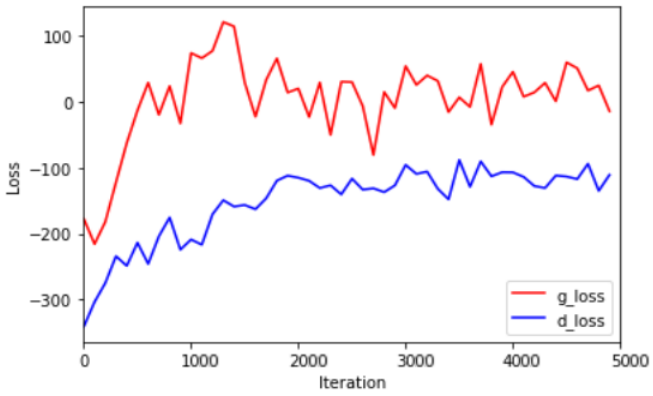
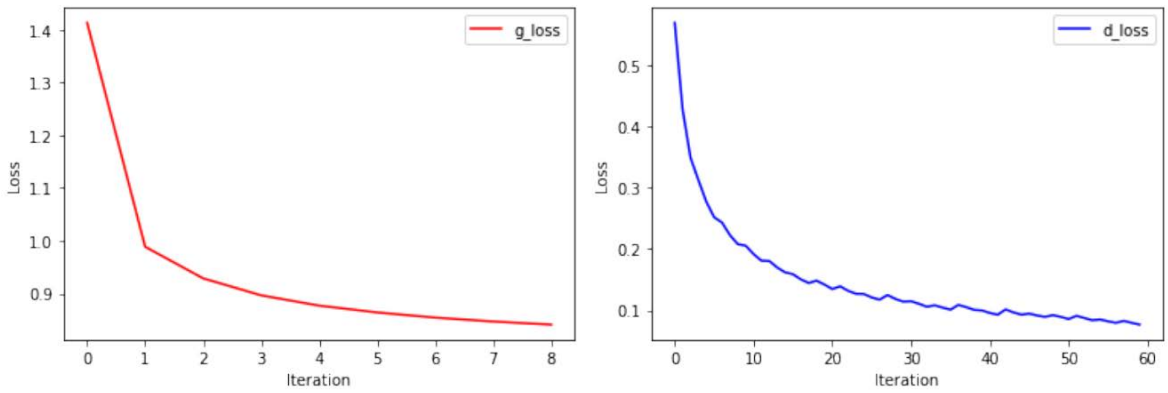


Fig.9. WGAN loss graph.

SeqGAN Model

The Generator of SeqGAN is pretrained using MLE for 20 epochs, while the Disciminator is pretrained for 10 epochs. After completing the pretrained phase for both Generator and Discriminator, SeqGAN is trained using Policy gradient for 3 steps. In each step, the Generator is trained first with Monte Carlo search for $N = 8$ times, then the Discriminator is trained for 3 epochs. The batch size for SeqGAN training is 64. The dimension of embedding and hidden for Generator is 32, and Discriminator is 64.

Figure 10a and Fig 10b respectively indicate g_loss and d_loss of SeqGAN model in each iteration. We see the phenomenon of convergence failure happened in SeqGAN training. d_loss rapidly decreases to a value close to zero at 60 epoch and remains this during training. This also happens with g_loss from 8 iteration.



(a) (b)

Fig.10. The loss graphs of the Generator (a) and Discriminator (b) in SeqGAN.

C. The Classification Results of the Discriminators in Different GAN Models

The trained GAN models are utilized for testing on both malign and benign URL samples in the test set.

Table 3. The classification accuracy of the the Discriminators in different GAN models.

Model	GAN	DCGAN	WGAN	SeqGAN
Accuracy (%)	99.98	99.88	98.99	85.05

Table 3 demonstrates the results of the Discriminator classifiers of different GAN models. We can see that basic GAN and DCGAN achieve the highest scores on discriminating the URLs, followed by WGAN. SeqGAN model has the worst performance. This comes from the fact that the models of GAN, DCGAN and WGAN have similar architectures. Therefore, their classification results are close to each other and better than SeqGAN. The Discriminators of GAN and DCGAN are similar in the way they output probabilities of whether the generated URL samples are similar to the target samples. WGAN Discriminator scores the generated URL samples and returns the Wasserstein distance as the output. This makes WGAN Discriminator has worse performance than GAN and DCGAN. In SeqGAN, the Discriminator provides a reward signal which guilds the generating action. However, the reward does not tell explicit whether a sample is fake or not. Thus, it makes SeqGAN's classifier performance worse than other GAN models.

The phishing URL classification results of the Discriminator networks in different GAN models in Table 3 show that the classification performance of the Discriminator network in the basic GAN model is the best. Although it is a basic model and its later variants such as DCGAN, WGAN or SeqGAN are created to improve the basic GAN model in the specific problems relating to generate new data patterns for training deep learning models. However, in this work, it is proven that just the basic GAN model can do the best for malicious URL detection.

4.3. Effective Evaluation of URL Strings Generated from the Generator Networks

In this section, the clean and phishing URL samples of the original dataset are utilized for training each GAN model. The purpose of this training is to generate the URL patterns that have structure as similar to the original ones as possible. This is measured by two indicators of TLD and SSIM scores. The efficiency of the GAN-generated URL patterns is also evaluated by their ability to fool the phishing URL classifiers of RF and LSTM. Two training scenarios of the classifiers are set for this. First, we train RF and LSTM classifiers on the original set of URL samples. Second, the URL samples generated by GAN are added to the original training set to form a new training set for these classifiers. The trained classifiers are then tested on the same test set to show the comparative results. The test set contains 1M benign and 1M malign URL strings.

A. Structural Similarity Evaluation between the Real and Fake URLs Generated by GANs

In order to evaluate the structure similarity between real and GAN-generated URL samples, we propose two evaluation criteria: (1) TLD-based score (Top-level Domain) and (2) SSIM score (Structural Similarity Index Score). The first one indicates that the fake URL strings generated from GAN models should contain TLD to be considered legitimate URLs. They otherwise will be considered illegitimate or phishing and do not present a URL standard structure. The url-lib package is used in this work to parse fake URL strings to several components for this checking.

Secondly, we evaluate the SSIM score of the GAN-based generated URLs to the real URLs. The SSIM is firstly used to measure the structural information of an image. It represents the dependencies among the pixels in an image. The SSIM score can be applied to textual strings to measure the interdependence of the characters in the URLs, thereby indicate the structural similarity of the URL strings. The SSIM score for two URL strings of **x** and **y** as follows:

$$SSIM(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy})}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \quad (8)$$

where μ_y denoted the mean value of the URL sequence y and σ_y^2 is the variance of the mentioned sequence. $c_1 = (Lk_1)^2$ and $c_2 = (Lk_2)^2$, with L is the dictionary size for character-level encoding of a URL string, $k1$ and $k2$ are set at 0.01 and 0.03 respectively for stabilizing the division with weak denominator of SSIM.

Table 4. The TLD and SSIM scores of different GAN models.

Metrics	GAN	DCGAN	WGAN	SeqGAN
TLD (%)	49.60	50.68	49.42	92.22
SSIM	0.49	0.02	0.67	0.48

Table 4 indicates the scores gained from the experiments on different GAN models. TLD scores of GAN, DCGAN and WGAN fluctuate slightly at about 49 to 50%. It could be explained by the nature of these three GAN models. They are designed to work with continuous data like generating pixel values from images. With discrete data such as text and character, these models' performance is worse than discrete models such as MLE (Maximum Likelihood Estimator) or SeqGAN. In perspective of SSIM, WGAN has the best SSIM score of 0.67 while SeqGAN and GAN have nearly same scores at 0.48 and 0.47, respectively. However the worst SSIM value belongs to DCGAN model, with only 0.02. These SSIM results are reflected in Table 5 and Table 6.

Table.5. Examples of benign samples generated by GAN models.

Model	**Benign Sample**
GAN	httʊs6w.aקả.лo*成品 ms*i×too?ņl/0re–ô
DCGAN	ht/-s:-h#ę>ɥe”lrвthлdbemиe:¤.cob ͺ
WGAN	https///ww.pur-kicirw.com/t.porg/phagene-ha
SeqGAN	https://idanicies.hh.phoowvonter/rang_1

Table.6. Examples of malign samples generated by GAN models.

Model	**Malign Sample**
GAN	v+品 iediblo½jf"torl.sצ-:è!/
DCGAN	xę"т®тėרö,¨£ţki“½¢тï€×:;©c2â§)nͺ
WGAN	rtlsleeliraleacosacehse.blogüele.cr
SeqGAN	speraniticesp.net

As observed in Table 5 and Table 6, the URL samples generated by SeqGAN have the structural form of canonical URLs, while GAN and DCGAN samples are much more noisy and unstable. In case of WGAN, the bad situation of GAN and DCGAN seem to be in remission. WGAN can learn the "https" prefix from the training data, but it can not learn the structural form of "https://". This means in a standard URL string, there are always the symbols of "://" after "https". As the illustrated example in Table 5, the correct URL structure should be "https://ww.pur-kicirw.com/t.porg/phagene-ha" instead of "https///ww.pur-kicirw.com/t.porg/phagene-ha" generated by WGAN.

B. Evaluation of the Ability to Deceive Phishing URL Classifiers by GAN-generated URL Samples

In order to evaluate the ability of deceiving the classifiers of the generated URL samples by GAN models, two training procedures are done: (1) train GAN models to generate new URL samples and (2) train the classifiers with the original dataset and the URL samples generated by GAN models.

GAN models are train on the balanced dataset of both malign and benign URL samples. Each GAN architecture is trained to learn separately from malign and benign URL strings. The dataset for these trainings is the original one with 1M for each malign and benign URLs. However, the scheme component in each clean URL string is removed to anti data bias. Because in the original dataset, 99.82% of the clean URLs contain prefix such as "https" while this is not included in the phishing data.

The classifiers are trained with two scenarios. The first one is to use 1M phishing and 1M clean URLs from the original balanced dataset (OBD). The second scenario is training on this original balanced dataset plus 1M GAN-generated URL samples for each type of malign and benign URLs (OBD+GANgen). The trained classifiers are then tested on the test set of 1M malign and 1M benign URLs. The classifiers of Random Forest and LSTM are utilized in this work to evaluate the effect of the GAN-generated URLs in deceiving the these phishing URL detectors. Random Forest model is trained with maximum depth of 10 with no random state. LSTM classifier is trained with embedding size of 128 and hidden size of 64. LSTM is trained with 5 epochs using Cross Entropy loss and Adam as the optimizer. The learning rate of Adam is set at 1e-3 with no weight decay.

Table 7. The classification results for two training scenarios of RF and LSTM on OBD and OBD+GANgen.

Classifier	**Classification Accuracy**				
	OBD	OBD+GANgen			
		GAN	DC GAN	WGAN	SeqGAN
Random Forest	90.46	90.85	90.82	90.48	**90.28**
LSTM	96.45	**96.16**	**96.39**	96.53	**96.23**

Table 7 presents the comparative classification results of two training scenarios for the classifiers of Random Forest and LSTM. The ability to fool the classifiers of different GANs is evaluated by the classification accuracy they have when training the classifiers on the original balanced data (OBD) compared to the case of traing on ODB plus URL data synthesized by each GAN model. In comparison with the case of OBD training, the lower the classification accuracy, the better the URL patterns generated by GANs in fooling the classifiers. In case of Random Forest, SeqGAN is the best with the lowest classification accuracy of 90.28% compared to other GAN models (90.48% of WGAN, 90.82% of DCGAN and 90.85%) and the lower accuracy in comparison with the OBD case of 90.46%. All models of GAN, DCGAN and WGAN achieve classification accuracy higher than OBD case. This demonstrates the ineffectiveness of the URL patterns generated by these GAN models in fooling the Random Forest classifier. In case of LSTM classifier, the effect of deceiving the classifier is achieved with URLs generated by GAN, DCGAN and SeqGAN models. Among them GAN is

the best one with 96.16% compared to 96.23% of SeqGAN, 96.39% of DCGAN and 96.45% of OBD. However, this does not happen with WGAN of 96.53% which is higher than OBD case.

From the above analysis of the classification results, it is shown that with different URL classifiers, the classifier fooling efficiency of URLs generated by different GANs will not be the same. SeqGAN can be the best one for deceiving Random Forest classifier but in case of LSTM classifier, GAN is the most effective model.

5. Conclusions

In this work, we explore both defending and attacking perspectives of GAN models on phishing URL detection and generating URL samples to fool URL classifiers/detectors. Unlike in image and text generation, the URL character sequence preserves some non semantic structures and degrees of noise, which is a challenge for the standard GAN models. Four different GAN models are evaluated in our works. Each targets different perspectives of the GAN capabilities. In defending perspective, the Discriminator of the standard GAN model has the highest accuracy among other tested approaches. In the four tested GAN methods, three classifiers based on the Discriminator can reach the accuracy of about 99.0 for classifying the benign URL samples. Meanwhile, in the attacking perspective, SeqGAN can simultaneously generate the most realistic URL samples as well as fool the phishing URL classifiers of RF and LSTM. The synthesized URs samples can affect the classification capacity of RF and LSTM to some extend. From our results, the GAN-based approach can be generalized to several main directions in URL phishing classification such as domain adaptation or online learning classifier.

References

[1] https://docs.broadcom.com/doc/istr-24-2019-en (Online: Nov 08, 2022).
[2] COFENSE: cofense-annual-report-2021.pdf [Online]. Available: https://cofense.com/wp-content/uploads/2021/02/.
[3] https://terranovasecurity.com/2021-gone-phishing-tournament-results/
[4] CISCO: 2021-cyber-security-threat-trends-phishing-crypto-top-the-list. [Online]. Available: https://umbrella.cisco.com/info/.
[5] https://datatracker.ietf.org/doc/html/rfc3986.
[6] Marchal, Samuel and Saari, Kalle and Singh, Nidhi and Asokan, N, “Know your phish: Novel techniques for detecting phishing sites and their targets,” in 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 323–333. Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
[7] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).
[8] Denton, Emily L., Soumith Chintala, and Rob Fergus. "Deep generative image models using a laplacian pyramid of adversarial networks." Advances in neural information processing systems 28 (2015).
[9] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.
[10] Pham, Tuan Dung, et al, "Exploring Efficiency of GAN-based Generated URLs for Phishing URL Detection," International Conference on Multimedia Analysis and Pattern Recognition (MAPR). IEEE, 2021.
[11] Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired image-to-image translation using cycleconsistent adversarial networks." In Proceedings of the IEEE international conference on computer vision, pp. 2223- 2232. 2017.
[12] Yu, Lantao, Weinan Zhang, Jun Wang, and Yong Yu. "Seqgan: Sequence generative adversarial nets with policy gradient." In Proceedings of the AAAI conference on artificial intelligence, vol. 31, no. 1. 2017.
[13] Zhang, Yizhe, Zhe Gan, and Lawrence Carin. "Generating text via adversarial training." In NIPS workshop on Adversarial Training, vol. 21, pp. 21-32. academia. edu, 2016.
[14] https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse.
[15] Moghimi, Mahmood, and Ali Yazdian Varjani. "New rule-based phishing detection method," Expert systems with applications 53 (2016), pp. 231-242.
[16] Singh, Charu. "Phishing Website Detection Based on Machine Learning: A Survey." 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, 2020.
[17] Sabir, Bushra, M. Ali Babar, and Raj Gaire. "An evasion attack against ml-based phishing url detectors." arXiv preprint arXiv:2005.08454 (2020).
[18] Kumar, N., Misra, S., Obaidat, M., ‘Collaborative Learning Automata-Based Routing for Rescue Operations in Dense Urban Regions Using Vehicular Sensor Networks’, In IEEE Systems Journal, DOI: 10.1109/JSYST.2014 22335451, 2014, pp 1081-1090.
[19] Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." Advances in neural information processing systems 29 (2016).
[20] Karras, Tero, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive growing of gans for improved quality, stability, and variation." arXiv preprint arXiv:1710.10196 (2017).
[21] Corley, Isaac and Lwowski, Jonathan and Hoffman, Justin, “Domaingan: generating adversarial examples to attack domain generation algorithm classifiers,” arXiv preprint arXiv:1911.06285, 2019
[22] Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock, “Character level based detection of dga domain names,” In 2018 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2018.
[23] AlEroud, Ahmed, and George Karabatis, “Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks, ” Proceedings of the Sixth International Workshop on Security and Privacy Analytics, 2020.
[24] Kamran, Sharif Amit, Shamik Sengupta, and Alireza Tavakkoli, "Semi-supervised Conditional GAN for Simultaneous Generation and Detection of Phishing URLs: A Game theoretic Perspective," arXiv preprint arXiv:2108.01852 (2021).

[25] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.
[26] Pham, Thuy Thi Thanh, Van Nam Hoang, and Thanh Ngoc Ha. "Exploring efficiency of character-level convolution neuron network and long short term memory on malicious url detection." Proceedings of the 2018 VII International Conference on Network, Communication and Computing. 2018.

Authors' Profiles

Thi Thanh Thuy Pham is main lecturer at Faculty of Information Security, Academy of People Security, Hanoi, Vietnam. She received an Engineering degree in Cryptography Techniques from Academy of Cryptography Techniques, Vietnam; a master's degree in Communication and Information Processing, and PhD degree in Computer Science from Hanoi University of Science and Technology. She is interested in machine learning and deep learning applied to network and information security; computer vision.

Pham Tuan Dung graduated from Vietnam National University, University of Engineering and Technology. With his Bachelor Degree in Computer Science, he continue to work in the Faculty of Engineering and Technology, UET as assistant lecturer and study his Master in Computer Science. His research interest include privacy learning with deep learning, generative models for text and images.

Viet Cuong TA completed his BSc. and MSc. degrees at University of Engineering and Technology, VNU Hanoi. He received his Ph.D degree from University of Grenoble Alpes in 2018 for topics on indoor positioning with multiple data source. Since then, he is a lecture at Faculty of Engineering and Technology, University of Engineering and Technology, VNU Hanoi. His main research interests are learning representation with deep networks, generative models and reinforcement learning.

How to cite this paper: Thi Thanh Thuy Pham, Tuan Dung Pham, Viet Cuong Ta, "Evaluation of GAN-based Models for Phishing URL Classifiers", International Journal of Computer Network and Information Security(IJCNIS), Vol.15, No.2, pp.1-14, 2023. DOI:10.5815/ijcnis.2023.02.01