Evaluating the effectiveness of Discriminator network in GAN architecture for phishing URL classification

Pham Thi Thanh Thuy[1], Ta Viet Cuong[2*]

[1]Faculty of Information Security, Academy of People Security;
[2]HMI lab, VNU University of Engineering and Technology.
[*]Corresponding author: cuongtv@vnu.edu.vn
Received 6 Jan 2023; Revised 27 Mar 2023; Accepted 10 Apr 2023; Published 28 Apr 2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.86.2023.110-119

ABSTRACT

Phishing attack by illegitimate URLs is of the most common security challenges for both individuals and companies in ensuring the security of their information resources. The user passwords, credit card information, or other sensitive information can be stolen by clicking on the malicious URL links. Recently, machine learning based approach is being popularly applied to detect phishing URLs. The classifiers, such as SVM, Random Forest, LSTM, etc., are built on the standard datasets to make a prediction about a URL sample is malign or benign one. Some recent researches focus on using GAN network for enrichment of malicious URL samples utilized in classifier training based on deep learning models. In this work, we explore the ability of training a standard GAN architecture which consists of two adversarial networks of Discriminator and Generator. The URL samples are generated by the Generator network will be refined and feed backed to the Generator by the Discriminator. This helps the Generator generate URL samples that are more and more similar to the real ones. Accordingly, the Discriminant network also learns the malicious and clean characteristics of the URL patterns. In order to evaluate the effectiveness of this learning, the experiments are conducted on completely new testing datasets beyond the training datasets. The experimental results are promising with the classification accuracy of both malign and benign URLs are about 97%.

Keywords: Phishing URL detection; GAN; Discriminator-based classification.

1. INTRODUCTION

Many applications and network services are now developed on the web platform. In web, along with the concept of hypertext and the web protocol HTTP (Hyper Text Transfer Protocol), URL is one of the important concepts of the Web. URL stands for Uniform Resource Locator. It is the address of a unique resource on the Web. Theoretically, each valid URL points to a unique resource. Because a resource is represented by a URL address, and the URL itself is handled by the Web server, the owner of the web server must carefully manage the resource and the URL assigned to it. However, nowadays, malicious or phishing URL links appear more and more and are rapidly shared through social networks. Just clicking on a phishing link can bring potential dangers to Internet users.

The creation of malicious URLs can be done by bad guys in many ways. A fake domain name can be injected to a URL; paths and file elements of a true URL string can be changed by will of the attackers to become the phishing one. In addition, an attacker can register any domain name that has not been registered before for phishing attack. Other common methods used by attackers are Cybersquatting and Typosquatting. Users can accidentally enter an incorrect website address or click on a link that looks like a trusted domain name and from there they could be redirected to the phishing sites.

On the user side, the users can identify a malicious links/URLs by detecting several strange signs such as missing information and typos; strange domain names; mimic the domain name of a legitimate website; long domain names. However, due to the natural of URLs and various techniques in crafting URL strings, it is difficult to detect such bad malign URL strings manually.

For automatic malicious URL detection systems, several solutions are proposed, and they can be categorized into three main approaches: (1) Blacklist-based method [1, 2]; (2) Heuristics approach [3]; and (3) machine learning-based method which recently has been attracted by research and application community. Many machine learning algorithms have been used for the URL classification. In general, there are two branches of machine learning algorithms: (1) traditional machine learning algorithms, such as SVM (Support Vector Machine), decision tree [4], random forest [5], and (2) deep learning using multi-layer neural network model.

In this work, a GAN-based deep learning model is proposed. It consists of 2 parts: (a) training two adversarial GAN networks of Discriminator and Generator; (b) URL classification based on the trained Discriminator. It is different from other general GAN-based approaches that pay attention on training the Generator to create fake URL patterns that look like the real ones, in this work, the Generator network is focused to enhance the classification ability of the Discriminator network. The use of Discriminator is viewed from an adversarial setting where the testing URLs are generated from a predefined distribution. By learning to classify the generated URLs, the Discriminator can generalize from training URL samples to testing URL samples. In addition, in this work, we focus directly on URL textual strings to exploit at character level for feature encoding and extraction. This treatment is much simpler than other multi-cue feature extraction approaches but still brings high performance in model training and classification. The experiments are done on large dataset of malign and benign URL samples. On the test set, the classification results are promising with 96.98% and 97.77% for malign and benign URL strings, respectively.

The remainder of the paper is organized as follows. Section 2 is analysed with some related works. Section 3 presents the proposed framework and GAN architecture. The detailed implementation of the proposed solution with the experimental evaluations are discussed in section 4. Finally, conclusion and future work are shown in section 5.

2. RELATED WORKS

The work on phishing URL detection and classification includes a wide range of approaches. In [6], the authors employ lexical-based features with linear weighted to classify a phishing URL. More complex models could be employed such as [4], Random Forest [5] and Extreme Gradient Boost [7].

Recently, many deep learning models are applied for URL feature extraction and classification, such as Convolutional Networks [8], Recurrent Networks [9, 10] and Generative Adversarial Networks [11]. In [12], a CNN-based architecture of VGG-16 and LSTM (Long Short Term Memory) are proposed for malicious URL detection. The experimental results are promising, especially for the fusion scheme of LSTM and CNN, with above 96% for precision and 98% for recall on the experimental datasets. However, the experimental datasets in this work are noisy with many duplicate URL patterns. This may affect the actual classification accuracy of the proposed models. The authors in [13] proposed a semi-supervised Conditional GAN for simultaneous generation and detection of phishing URLs. The discriminator's accuracy for classification of benign or malign URLs is 95.52%.

In [14], a framework consists of two phases, namely training the GAN and predicting using GAN is proposed. Feature extraction is done based on the internal structure and external metadata of a website. The GAN training is done in two ways: (1) giving it clean data and train it to detect clean websites and (2) doing the opposite by giving it phishing data, train it to detect phishing websites. The prediction phase using the trained GAN is done on the test set which takes up 10% of the dataset, while 90% of the dataset is used for GAN training phase. The accuracy of the GAN trained with the clean dataset is significantly more efficient than the GAN trained with the phishing dataset, with 94% and 91.96%, respectively. Compared to previous related works, in this paper, we provide a comprehensive study on applying the discriminator of GAN for classifying the malign URL strings directly.

3. THE PROPOSED FRAMEWORK

The proposed framework consists of two parts: (1) training a GAN model with two adversarial networks of Discriminator and Generator; (2) URL classification by the Discriminator of the trained GAN. Figure 1 shows the components of the proposed framework. Starting from an input URL, an embedding layer is used to convert it into the feature domain which can work with the Discriminator and Generator of GAN model. The Discriminator receives an URL string, which is encoded in the feature domain and outputs a probability distribution. The output value is used with different meaning in training and testing phases. In the training phase, the Discriminator attempts to classify the real input URLs and the synthetic URLs. The synthetic URLs are generated from noise space by the Generator of GAN. In the testing phase, the Discriminator receives an input as an encoded URL string and outputs a classified value of malign or benign URL.

Instead of general approaches of GAN which focus on training the Generator, our proposed framework use the Generator to strengthen the classification capacity of the Discriminator. We rely on the assumption that each generated URL could be considered as a malign URL. Unlike in other text generated tasks, the latent space of real URL is not easy to capture. Our work mainly examines how complex the latent space is, given a set of real and fake URLs. One of the main challenges in training GAN for URL classification is the model collapse mode, in which the Discriminator learns too fast thus prevents the Generator generates the appropriate samples. The proposed GAN architecture will be helpful in reducing the effects of collapse mode in training GAN.

3.1. URL embedding

In order to train and test the proposed model, an important preprocessing step that needs to be applied to the URL data samples is to convert a URL string of characters to a numeric vector. This conversion is called URL embedding or URL encoding.

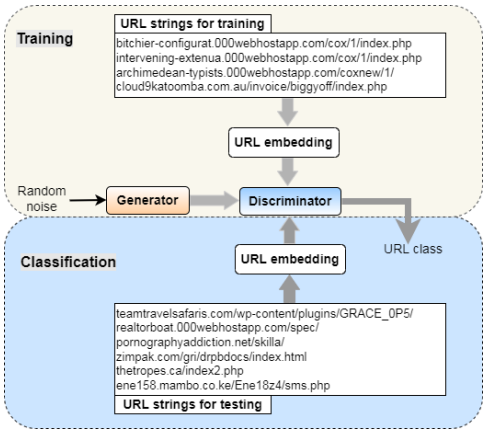

Figure 1. *Discriminator training and URL classification.*

```
'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f':
6, 'g': 7, 'h': 8, 'i': 9, 'j': 10, 'k': 11,
'l': 12, 'm': 13, 'n': 14, 'o': 15, 'p': 16,
'q': 17, 'r': 18, 's': 19, 't': 20, 'u': 21,
'v': 22, 'w': 23, 'x': 24, 'y': 25, 'z': 26,
'0': 27, '1': 28, '2': 29, '3': 30, '4': 31,
'5': 32, '6': 33, '7': 34, '8': 35, '9': 36,
'/': 37, ':': 38, '.': 39, '_': 40, '-': 41,
'(': 42, ')': 43, '=': 44, ';': 45, '?': 46,
```

Figure 2. *A dictionary of URL character.*

Given a set $U = \{(u_1, y_1), .., (u_i, y_i)\}$ with $(0 \leq i \leq n)$, where u_i is the i^{th} 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). In order to perform URL embedding, each URL string will be scaled to a set of m characters. Thus, each URL u_i consists of m characters 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. The dictionary D is established based on statistic of the unique characters that appear in the experimental database, including lowercase letters, numbers, and characters allowed in URLs (eg /, &, =...) as shown in figure 2. Figure 3 shows the number of occurrences of these characters in the database. Based on this statistic, we build a dictionary of size d=198.

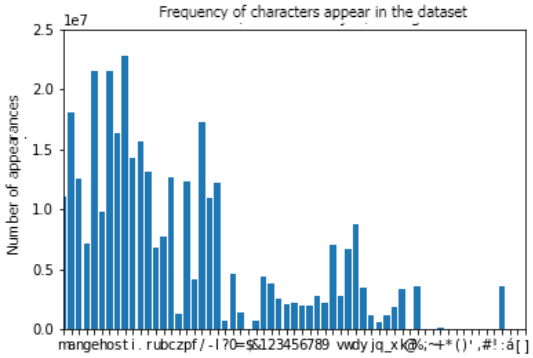

Figure 3. *Statistics of occurrence frequency of theunique characters in the dataset.*

Each URL string has a fixed length of m characters. Each encoded character generates an input tensor of fixed length of m. For the URL strings have the number of characters less than m, zero padding is added at the end of the strings to get the sufficient length of m. The size of the URL embedding vector will be ($m \times d$) (figure 4).

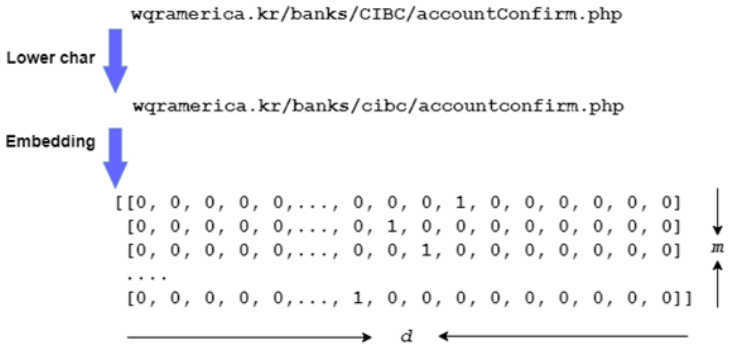

Figure 4. *URL embedding.*

Figure 5 indicates the analysis of the length distribution of the URL strings. From this, we choose a uniform size of 200 characters ($m = 200$) for a URL string.

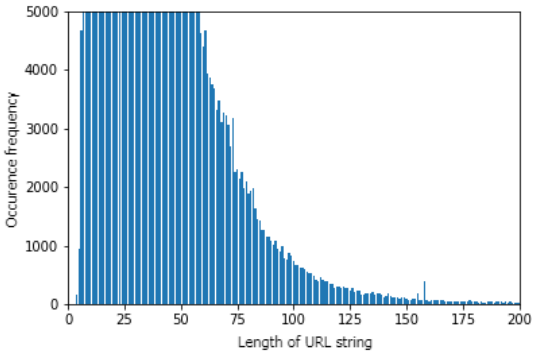

Figure 5. *URL length distribution according to the number of the characters in a URL string.*

3.2. Discriminator and Generator architectures

Figure 6 shows the detailed network architectures of the Discriminator and Generator. Discriminator network includes: (1) The input is a matrix representing the URL with dimensions $m \times d$. Then using Flatten to straighten this matrix into an array of $m \times d$ elements; (2) The FC layer, also known as Dense layer. The output dimension of this layer is 8 units. Using Dropout in this layer; (3) The FC layer with the output of this layer being 1 unit; (4) The FC layer with the output is 4 units. Using Dropout in this layer; (5) The FC layer gives the output of 1 unit; (6) The FC layer with the output of 2 units. Using Dropout in this layer; (7) The FC layer with the output of this layer is 1 unit. The activation function used is the Sigmoid function. The outputs are values in the range of [0-1].

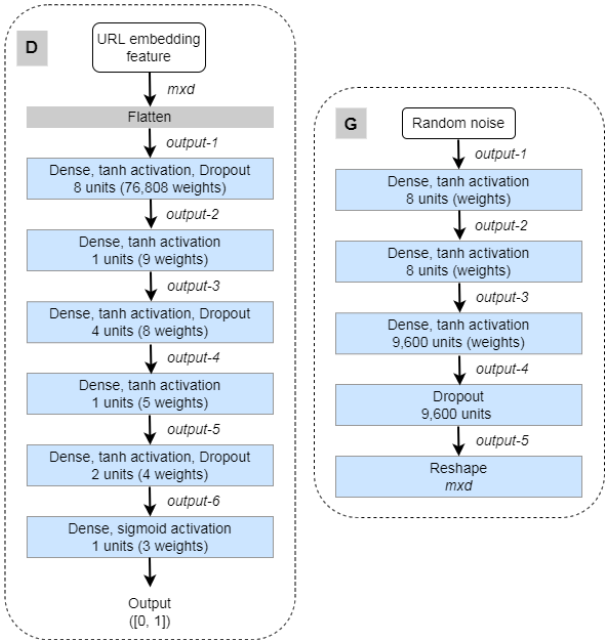

Figure 6. *Detailed network architectures of the Discriminator and Generator.*

The Generator network contains: (1) The input is a noise vector of size 8 units; (2) The FC layer gives the output of 8 units. This layer uses tanh activation function; (3) The FC layer with the output of 8 units and tanh activation function is used in this layer; (4) The FC layer brings to the output of 9,600 units, and uses tanh activation function; (5) Dropout is done to give the output of 9,600 units; (6) Reshaping to $m \times d$ matrix.

3.3. GAN training

In the GAN architecture, there are two networks of Generator and Discriminator that are trained in alternating periods. This means in the process of training GAN, the role of the Discriminator is: (1) to learn how to distinguish a URL sample generated by the Generator network is real or fake, and (2) to tell the Generator that the URL sample synthesized by the Generator network does not fool the Discriminator and needs to be improved. Based on this, the Generator will make the URL sample more similar to the real one and the Discriminator will also become proficient at distinguishing between fake and real URL sample. The expectation from training GAN is the fake URL strings can deceive the Discriminator.

Let's denote a set of fake and real URLs as U. Given a real URL (U_real) and a fake URL (U_fake), the Discriminator, which plays the role of a binary classifier, learns to discriminate between a real and fake URL samples. In other words, the Discriminator tries to determine a URL belongs to the true data distribution P_data of the training set or the model distribution

P_model of the URLs generated by Generator. First, the URL patterns generated by the Generator U_fake and the original URLs U_real in the training set are passed to the Discriminator. The Discriminator then predicts Y_pred (a probability score) which shows the U samples are real or fake.

Next, the predictions are compared with the ground truth (0: fake, 1: real), and a Binary Cross-Entropy loss is calculated. The loss is then backpropagated only through the Discriminator, and its parameters are optimized accordingly. Second, the Generator produces URLs ($\boldsymbol{U}_{fake}$), which are again passed through the Discriminator. Here too it outputs a prediction $\boldsymbol{Y}_{pred}$ and the Binary Cross-Entropy loss is computed. Now, in this alternate step, because we want to enforce the Generator to produce URL samples as similar to the real URLs as possible, the true labels (or ground truth) are all labeled as 'real' or 1. As a result, when the Generator tries to fool the Discriminator (into believing that the URL samples generated by it are real), the loss is backpropagated only through the Generator network, and its parameters are optimized suitably.

The Binary Cross-Entropy loss function is calculated as follows:

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i log\hat{y}_i + (1 - y_i) log\ (1 - \hat{y}_i)] \quad (1)$$

where N is training samples per batch, $\hat{y}_i$ is the prediction made by the Discriminator model in GAN, and y_i is the true label. This Binary Cross-Entropy loss function is modeled in GAN as follows:

- The Discriminator is a binary classifier, given an input u, outputs a probability $D(u)$ between 0 and 1. As the true label for $\boldsymbol{U}_{real}$ is 1 and the true label for $\boldsymbol{U}_{fake}$ is 0, so the probability $D(u)$ closer to 1 means that the Discriminator predicts the input to be a real URL, and a probability closer to 0 means that the input is fake URL. Thus, the objective of the Discriminator becomes: (1) Maximizing the probability $D(U_{real})$ means bring it closer to 1; (2) Minimizing the probability $D(\boldsymbol{U}_{fake})$, where $\boldsymbol{U}_{fake}$ is G(Z). For the first objective of the Discriminator, the true label y is 1, and the predicted output $\hat{y}$ is $D(U_{real})$. Putting these values in the BCE loss function, we get $D_{loss_{real}} = -logD(\boldsymbol{U}_{real})$. For the second objective of Discriminator, the true label y is 0, and the predicted output $\hat{y}$ is $D(\boldsymbol{U}_{fake})$. where $\boldsymbol{U}_{fake}$ is equal to $G(z)$. Putting these values in the BCE loss function, we have $D_{loss_{fake}} = -log(1 - D(G(z))$. Therefore, the cumulative Discriminator loss becomes:

$$\boldsymbol{D}_{loss} = -log\boldsymbol{D}(\boldsymbol{U}_{real}) - log\ \left(1 - \boldsymbol{D}(\boldsymbol{G}(z))\right) \quad (2)$$

- The generator wants the URLs generated by it to be classified as real by the Discriminator. Thus, the objective of the Generator is to maximizing the probability $\boldsymbol{D}(\boldsymbol{G}(z)$. This means the true label y is 1, and the predicted output $\hat{y}$ is $D(G(z))$. Putting these values in the BCE loss function, we get:

$$\boldsymbol{G}_{loss} = -log\boldsymbol{D}(\boldsymbol{G}(z)) \quad (3)$$

After training the Discriminator network, we will use this network to classify the URLs in the new test sets.

4. EXPERIMENTS AND RESULTS

This section shows the settings for experimental data and model parameters. The experimental results are shown with two cases: (1) training GAN with malign URL strings and testing the Discriminator classification on the test set of clean and phishing URLs; (2) training GAN with benign URL strings and testing as the same of (1).

4.1. Data and the model parameters

In this work, the experimental dataset contains benign and malign URL strings. The benign URLs are collected from Commoncrawl1, with more than 3 million URL samples. Common Crawl is a freely available dataset which contains over 10 years of crawled data including over 25 billion websites, trillions of links, and petabytes of data. The malign URL strings are utilized in this work are the ones of [15], with more than 3 million URL samples. In [15], the dataset of malicious URLs is aggregated from several trusted sources such as Ransomware Tracker, Google SafeBrowsing API, Cisco Umbrella, Virus Total API, Panda security, Open Fish, Kaggle Data, Adblock, Pi-Hole porn block. The collected URL samples are then gone through data preprocessing, error removal, junk data filtering and data type normalization. In this dataset, the URLs identified as malicious include 5 main types of spam, phishing, malware, ransomware and darkweb.

The experimental dataset will be splited into two parts: one for model training (3 million samples for each URL type of malign and benign); another part is used for model testing (200,000 samples for each URL type).

In order to train GAN model, the input of URL string is encoded into a $m \times d$ matrix. GAN models are trained separately for each class of malign and benign URLs in the training dataset. The trained Discriminators will then be used to classify URLs in the test set. The test set is the set of new URL strings, which are not included in the training dataset. The training is done with batch size of 128, epoch of 1000, using Adam optimal algorithm with learning rate of 0.001 and dropout of 0.5. Generator and Discriminator models are trained separately.

4.2. Experimental results

4.2.1. The evaluations for the proposed framework

- Training results on malign URLs

Figure 7a shows the Discriminator and Generator loss values through each iteration. It can be seen that they are quite stable, ranging from 0.50-0.75 for Discriminator network (d_{loss}) and from 0.75 to less than 1.25 for Generator network (g_{loss}). Figure 7b further shows the Discriminator loss values for the real and fake samples. For real samples, the loss value fluctuates around a threshold of approximately 0.50, while for fake samples, this value ranges from about 0.6 to 1.25.

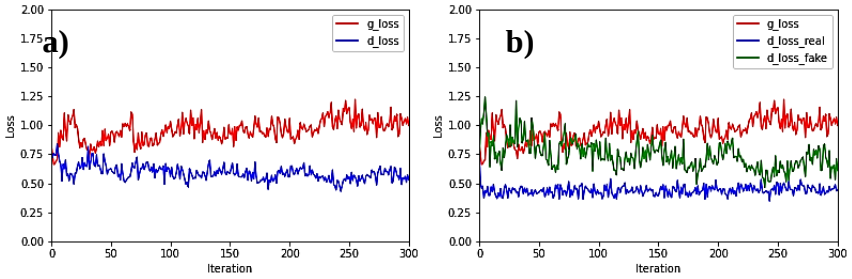

Figure 7. *Discriminator Loss (d_{loss}) and Generator Loss (g_{loss}): a) For all train dataset; b) For training with real and fake samples in training with malign URL strings.*

Figure 8 shows that the capability to distinguish between real and fake samples of the Discriminator network improves as the training progress. At the several starting iterations, it cannot differ between the real and the fake samples. After that, it increases gradually and start to converge at around 200 iterations. When the training reach 300 iterations, it can perfectly discriminate the two type of URLs.

After training the Discriminator on the malicious URLs, the performance of this training is test

1 https://index.commoncrawl.org

on two test datasets of 200K phishing URLs and 200K clean URLs. With a threshold set for classification of phishing and clean URLs of 0.8, the classification accuracy of 200K phishing URLs is 96.98% (figure 9a), while with 200K clean URLs, this score is below the threshold (figure 9b).

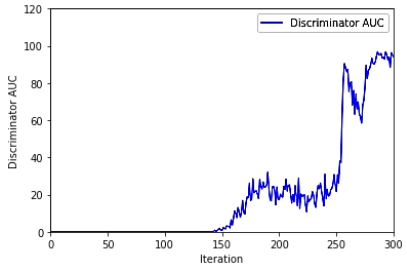

Figure 8. *Discriminator area under the curve (AUC) when training with phishing URLs.*

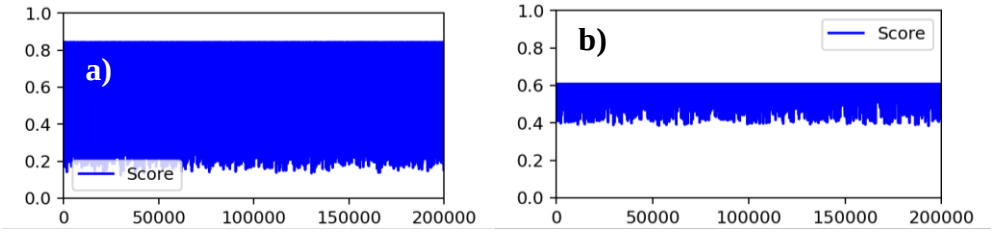

Figure 9. *The classification accuracy of Discriminator trained on malicious URLs when testing with 200K test samples of phishing URLs (a) and clean URLs (b).*

- Training results on malign URLs

Figure 10a shows the Discriminator and Generator loss values through each iteration. It can be seen that they are quite stable, ranging from 0.50 to 0.75 for Discriminator network (d_{loss}) and from 0.75 to approximately 1.0 for Generator network (g_{loss}). Figure 10b further shows the Discriminator loss values for the real and fake samples. For real samples, the loss value fluctuates around a threshold of approximately 0.50, while for fake samples, this value ranges from about 0.5 to 1.0. Figure 11 shows that the capability to distinguish between real and fake samples of the Discriminator network improves as the training progress. At about the first 70 iterations, it cannot differ between the real and the fake samples. After that, it starts converge at later iterations. When the training reach 300 iterations, it can perfectly discriminate the two type of URLs.

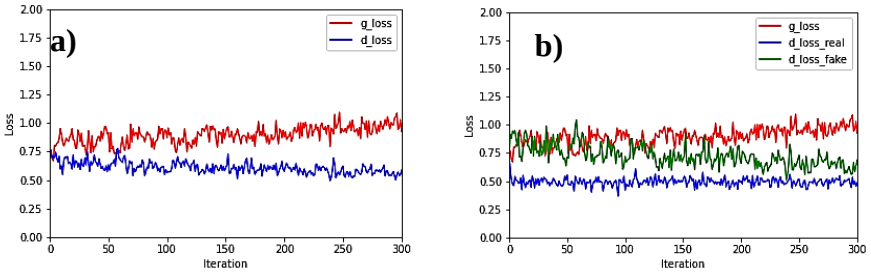

Figure 10. *Discriminator Loss (d_{loss}) and Generator Loss (g_{loss}): a) For all train dataset; b) For training with real and fake samples in training with malign URL strings.*

After training the Discriminator network on the clean URL strings, we test it on two test URL datasets including 200K malign URLs and 200K benign URLs. Also with the threshold set for classifying phishing and clean URLs as 0.8, we get the classification accuracy for 200K clean URLs is 97.77% (figure 12a) while with 200K phishing URLs the score is below the threshold (less than 0.6) (figure 12b). These testing results suggest that the Discriminator network has been trained effectively on the clean URL set, so that it can better classify the benign and malign URL samples in the testing dataset.

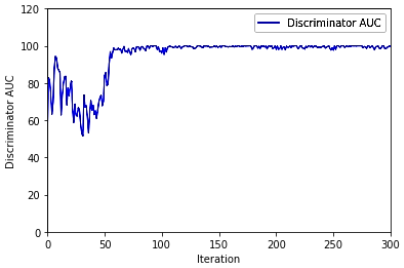

Figure 11. *Discriminator area under the curve (AUC) when training with phishing URLs.*

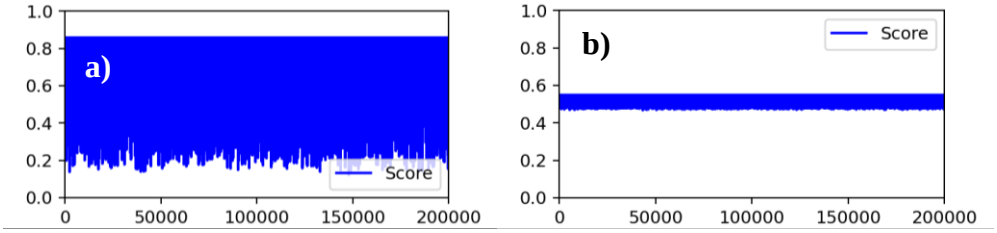

Figure 12. *The classification accuracy of Discriminator trained on clean URLs when testing with 200K test samples of clean URLs (a) and phishing URLs (b).*

4.2.2. The comparative evaluations with other methods of RF and XGBoost

In order to compare the performance of the proposed framework with two tree-based baseline methods of RF (Random Forest) and XGBoost (Extreme Gradient Boost), we train each model of RF and XGBoost on 22 features extracted from URL strings. It is a subset of the all features given in [16]. The AUC scores on 400K benign and malign from testing dataset with 3 models (our model, RF, XGBoost) are shown in table 1. It can be seen from the table that at 200 and 300 iterations, the AUC scores of our method are higher than both RF and XGBoost. At 200 iteration, it is 0.62 but at 300 iteration, it increases to 0.97, while the ones of RF and XGBoost are only 0.55 and 0.50, respectively.

Table 1. *AUC scores on 400K benign and malign from testing dataset with our model and other models of RF and XGBoost.*

Model	Iteration	AUC score
Random Forest	-	0.55
XGBoost	-	0.50
Our Method	200	0.62
	300	0.97

5. CONCLUSIONS

In this paper, we present an experimental investigation of URL classification by the Discriminator model in an standard GAN architecture. Training both Discriminator and Generator networks in alternative way by putting the original URLs and the generated URL samples that synthesized by the Generator will make the Discriminator becomes stronger for classifying benign and malign URL strings from the test sets. In the future works, the experiments are extended with the other GAN models, such as Conditional GAN (CGAN), Deep Convolutional GAN (DCGAN) or Wasserstein GAN (WGAN) to explore more about the game theoretic approach of the attacker (the Generator) and the Defender (the phishing URL detector of Discriminator).

REFERENCES

[1]. R. Srinivasa Rao, A. R. Pais, "*Detecting phishing websites using automation of human behavior*", in: Proceedings of the 3rd ACM workshop on cyber-physical system security, pp. 33–42, (2017).

[2]. C. L. Tan et al., "*Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder"*, Decision Support Systems 88, pp. 18–27, (2016).
[3]. D. L. Cook, V. K. Gurbani, M. Daniluk, "*Phishwish: a stateless phishing filter using minimal rules*", in: International conference on financial cryptography and data security, Springer, pp. 182–186, (2008).
[4]. L. Xu, Z. Zhan, S. Xu, K. Ye, "*Cross-layer detection of malicious websites"*, in: Proceedings of the third ACM conference on Data and application security and privacy, pp. 141–152, (2013).
[5]. B. Eshete, A. Villafiorita, K. Weldemariam, '*Binspect: Holistic analysis and detection of malicious web pages",* in: International conference on security and privacy in communication systems, Springer, pp. 149–166, (2012).
[6]. A. Blum, B. Wardman, T. Solorio, G. Warner, "*Lexical feature based phishing url detection using online learning"*, in: ACM Workshop on Artificial Intelligence and Security, pp. 54–60, (2010).
[7]. Madhu Chandra, S., K. T. Chandrashekar. "*Malicious url detection using extreme gradient boosting technique*", International Research Journal of Modernization in Engineering Technology and Science, Volume:02, Issue:10, pp. 675-682, (2020).
[8]. J. Saxe, K. Berlin, "*Expose: A character-level convolutional neural network with embeddings for* detecting malicious urls", file paths and registry keys, arXiv preprint arXiv:1702.08568.
[9]. P. Yang, G. Zhao, P. Zeng, "*Phishing website detection based on multidimensional features driven by deep learning"*, IEEE access 7, pp. 15196–15209, (2019).
[10]. Y. Huang, Q. Yang, J. Qin, W. Wen, "*Phishing url detection via cnn and attention-based hierarchical rnn"*, in: 18th IEEE International Conf. On TrustCom/BigDataSE, pp. 112–119, (2019).
[11]. A. AlEroud, G. Karabatis, "*Bypassing detection of url-based phishing attacks using generative adversarial deep neural networks"*, in: Proceedings of the Sixth International Workshop on Security and Privacy Analytics, pp. 53–60, (2020).
[12]. T. T. T. Pham, V. N. Hoang, T. N. Ha, "*Exploring efficiency of character-level convolution neuron network and long short term memory on malicious url detection"*, in: Proceedings of the 2018 VII International Conference on Network, Communication and Computing, pp. 82–86, (2018).
[13]. S. A. Kamran, S. Sengupta, A. Tavakkoli, "*Semi-supervised conditional gan for simultaneous generation and detection of phishing urls: A game theoretic perspective"*, arXiv preprint arXiv:2108.01852.
[14]. P. Robic-Butez, T. Y. Win, "*Detection of phishing websites using generative adversarial network"*, in: IEEE International Conference on Big Data. pp. 3216–3221, (2019).
[15]. H. V. Chi, "Xây dựng cơ sở dữ liệu huấn luyện phục vụ phát hiện URL độc hại", http://www.antoanthongtin.vn/gp-atm/ (2020) (in Vietnamese).

TÓM TẮT

Đánh giá hiệu quả của Discriminator trong kiến trúc GAN đối với phân loại URL độc hại

Tấn công lừa đảo bằng các URL bất hợp pháp là một trong những thách thức an toàn thông tin phổ biến nhất đối với cả cá nhân và tổ chức. Gần đây, phương pháp tiếp cận dựa trên học máy đang được áp dụng phổ biến để phát hiện các URL độc. Các bộ phân lớp như SVM, Random Forest, LSTM,... được xây dựng trên các bộ dữ liệu tiêu chuẩn để đưa ra dự đoán một mẫu URL là độc hay không độc. Một số nghiên cứu gần đây tập trung vào việc sử dụng mạng GAN để làm phong phú các mẫu URL độc hại được sử dụng trong huấn luyện bộ phân lớp dựa trên các mô hình học sâu. Trong bài báo này, chúng tôi khám phá khả năng huấn luyện một kiến trúc GAN với hai mạng đối nghịch là Discriminator và Generator. Các mẫu URL được tạo ra bởi mạng Generator sẽ được mạng Discriminator tinh chỉnh và phản hồi cho Trình tạo. Điều này giúp Trình tạo tạo ra các mẫu URL ngày càng giống lại so với mẫu thực. Theo đó, mạng Discriminator cũng học được các đặc trưng không độc và độc của các mẫu URL. Để đánh giá hiệu quả của việc huấn luyện này, các thử nghiệm được tiến hành trên các bộ dữ liệu thử nghiệm hoàn toàn mới so với bộ dữ liệu huấn luyện. Các kết quả thử nghiệm đầy hứa hẹn với độ chính xác phân loại của cả URL độc và không độc là khoảng 97%.

Từ khóa: Phát hiện URL độc hại; GAN; Phân lớp dựa trên mạng Discriminator.