Computers, Materials & Continua
DOI:10.32604/cmc.2021.015452
Article

Tech Science Press

Adversarial Attacks on Featureless Deep Learning Malicious URLs Detection

Bader Rasheed[1], Adil Khan[1], S. M. Ahsan Kazmi[2], Rasheed Hussain[2], Md. Jalil Piran[3,*] and Doug Young Suh[4]

[1]Institute of Data Science and Artificial Intelligence, Innopolis University, Innopolis, 420500, Russia
[2]Institute of Information Security and Cyberphysical Systems, Innopolis University, Innopolis, 420500, Russia
[3]Department of Computer Science and Engineering, Sejong University, Seoul, Korea
[4]Department of Electronics Engineering, Kyung Hee University, Yongin, Korea
[*]Corresponding Author: Md. Jalil Piran. Email: piran@sejong.ac.kr
Received: 22 November 2020; Accepted: 19 January 2021

Abstract: Detecting malicious Uniform Resource Locators (URLs) is crucially important to prevent attackers from committing cybercrimes. Recent researches have investigated the role of machine learning (ML) models to detect malicious URLs. By using ML algorithms, first, the features of URLs are extracted, and then different ML models are trained. The limitation of this approach is that it requires manual feature engineering and it does not consider the sequential patterns in the URL. Therefore, deep learning (DL) models are used to solve these issues since they are able to perform featureless detection. Furthermore, DL models give better accuracy and generalization to newly designed URLs; however, the results of our study show that these models, such as any other DL models, can be susceptible to adversarial attacks. In this paper, we examine the robustness of these models and demonstrate the importance of considering this susceptibility before applying such detection systems in real-world solutions. We propose and demonstrate a black-box attack based on scoring functions with greedy search for the minimum number of perturbations leading to a misclassification. The attack is examined against different types of convolutional neural networks (CNN)-based URL classifiers and it causes a tangible decrease in the accuracy with more than 56% reduction in the accuracy of the best classifier (among the selected classifiers for this work). Moreover, adversarial training shows promising results in reducing the influence of the attack on the robustness of the model to less than 7% on average.

Keywords: Malicious URLs; detection; deep learning; adversarial attack; web security

1 Introduction

Recent cyber-attacks have spurred an increased interest in devising security solutions to circumvent the threats posed by cyber attackers. It is, at least in part, due to the critical information

leakage as a result of attacks such as identity theft, Denial of Service (DoS), masquerading, impersonation, and so on. The attackers attempt to impersonate authorized users to steal important and critical information such as passwords, secret keys, and other personal information including bank account details. These attackers use any possible and available mediums to attract victims such as distributing impressive ads on the Internet, including malicious URLs in informative emails, or hacking a website. Such threats are collectively referred to as phishing which is a type of threat to sensitive information or data where attackers intentionally attack a victim [1]. The attacker lures the victim to a phishing webpage using different mediums and waits for the victim to access the phishing webpage (collectively referred to as social engineering approaches). According to the anti-phishing working group (APWG)'s newest report [2], in the first quarter of 2020, 165,772 phishing sites were detected on the Internet. Besides, 75% of these malicious sites use secure socket layer (SSL) protection which implies that it is not enough to rely only on SSL against such attacks.

Malicious URLs detection is a highly challenging task since there are no rules for generating URLs and the behavior of the URL must be studied to detect potential malicious URLs. Most of the existing traditional detection systems use database-oriented solutions such as blacklists, or heuristic-oriented solutions such as content or visual-based detection [3]. The URL-based techniques are safer and more realistic from three perspectives, i.e., no need to access the malicious webpage for performing dynamic analysis, the ability to perform Zero-Hour threat detection (i.e., for newly created websites), and reducing the amount of work and time to process a webpage compared to other existing approaches.

With the tremendous development in the field of ML in general and DL models in particular, these models are used to solve a wide variety of tasks in computer security and other fields [4–6]. ML helps in detecting any offensive events including detection of spam content, DoS attacks [7], attacks on industrial control systems (ICS) [8], attacks on Internet of Things (IoT) devices [9], malware detection, and malicious URL detection [10,11]. During the training process, ML models learn a prediction function that is used to classify a URL and then the trained model is used to classify new unseen URLs. For this purpose, we need a training dataset consisting of benign and malicious URLs. Furthermore, the training dataset should have informative features such that they adequately characterize the URL and that the benign and malicious URLs have different distributions. These features are usually extracted by domain experts and may include lexical features (such as the length of the URL, the existence of some words, bag of words, n-grams, etc.), and host-based features (Domain name properties, the geographical location of the host, etc.) [3]. To this end, different classification algorithms such as support vector machine (SVM), logistic regression, and decision tree classifiers can be used over the training data to learn the prediction function. However, extracting informative features is essential for the success of any classification model training. On the other hand, DL does both feature learning and prediction within the same model; hence, deep neural networks (DNNs) have the ability to discover the required secret features and use them to find a model that maps the input data to the desired output without explicitly defining the features by the domain experts. The first layers of the networks discover informative features and the latter layers use these features to make decisions [12]. In natural language processing (NLP), for instance, the model classifies a sentence according to the existence of some keywords. Hence, featureless malicious URL detection uses DL to classify a URL as benign or malicious [10]. The URL should be considered as a sentence and use the same approach used in NLP to classify it. This approach can eliminate the limitation of typical blacklist-based solutions [11] which cannot be generalized to novel URLs that do not exist in the blacklist.

Another advantage of DL-based featureless malicious URL detection systems is that they do not require feature engineering because they can extract the features automatically.

Nevertheless, despite remarkable results in providing intelligent solutions in different domains, ML and DL systems have shown more susceptibility to adversarial attacks in the form of small purposely created perturbations leading to misclassifications [13]. These attacks can cause dire consequences, especially when developing deep learning-based solutions for security-related problems where attackers work hard to discover new attack vectors. Evaluating the performance of these systems from only accuracy standpoint is not enough to decide if they are suitable for real-life applications or not. Since the purpose of applying these techniques is to protect against malicious activities, they cannot be applied until the robustness is considered to prevent the attacker from developing adversarial samples easily with small changes to the input sample.

To fill the gaps, the goal of this paper is three-fold: To highlight the existence of a vulnerability in featureless DL-based malicious URL detection systems that can be used by attackers to launch adversarial attacks on such systems, to develop a proof-of-concept algorithm to launch three attacks (character-based, segment-based, and full attack) and test these attacks on three kinds of classifiers (character-based, word-based, and full joint classifier), and to show how adversarial learning can be used to mitigate the threat of such attacks by augmenting the training data to effectively defend against such attacks.

The main contributions of this paper are summarized below.

- A novel attack on featureless deep learning-based malicious URL classification systems is introduced. The attack exploits the sensitivity of these systems to small input manipulation causing malicious URL misclassification as a benign URL. This attack works in a greedy mode providing the required perturbation with the least possible number of steps.
- CNN-based URL classifiers (character-based, word-based, and optimized joint classifier) are implemented and the performance of these classifiers is evaluated under our proposed attack.
- The ability of adversarial training is examined to defend against the proposed attack where the usability of this attack is demonstrated as a way for data augmentation to get more malicious URLs for training both accurate and secure DL-based URL classification systems.

The remainder of this paper is organized as follows. Section 2 presents related works in malicious URL detection with deep learning and adversarial attacks on text data. In Section 3, the problem of malicious URL detection is presented as a binary classifier along with a background of adversarial attacks. In Section 4, the attack is presented to fool URL classifiers and then in Section 5, the results of experiments on the robustness of three classifiers against the proposed attack are presented as well as the effectiveness of the proposed adversarial attack along with adversarial training as a defense against such attacks, are discussed. We discuss why these systems are vulnerable to adversarial attacks in Section 6 and provide some suggestions on how to make them secure. The limitations of this work are described in Section 7 and we conclude this work in Section 8.

2 Related Work

In this section, we review the existing works in malicious URL detection using deep learning and adversarial attacks on text data.

Malicious URL classification is a well-studied problem as malicious websites are a primary source of undesirable content and their timely detection is a crucial task. Recently, deep learning has been extensively used for malicious URL classification. In this regard, NLP-based deep learning models have been successfully applied for this task due to their ability to recognize semantic features from unstructured text data. For instance, eXpose [14] used character-level convolutional networks to classify a URL sample. The convolutional network tries to locate informative patterns of specific characters appearing together in the URLs. Because the model does not use word encoding, it does not have an explosion problem when increasing the number of features. The explosion problem appears when using word encoding because the vocabulary size of URLs words is unlimited with the ability to add new unseen words in each new URL. In addition to solving the manual feature extraction issue, eXpose outperforms manual feature extraction based URL classification models. Similarly, URLNet [10] applied convolutions for both characters and words in the URL sample to learn the URL final optimized embedding. Therefore, the model can have the ability to discover several types of semantic features of the URL. To solve the problem of large vocabulary size, URLNet uses an additional character-level word embedding where the final word embedding is created from the word itself and the characters present in that word. Furthermore, Shima et al. [15] used an advanced embedding method by embedding the combination of two characters appearing sequentially then CNNs are applied to classify the sample. On the other hand, in this work, we apply a full featureless approach for URL classification without any feature engineering, and we deal with rare unknown words by using word-embedding techniques.

The research in adversarial attacks on DL is more active on image data [16] than on texts. Consequently, most of the works in the adversarial text field try to use methods from the image field and apply them to texts. Samanta et al. [17] used the concept of fast gradient signed method (FGSM) to find and replace the important words or salient words that significantly affect the resulting class of the text when they are removed. Similarly, Sato et al. [18] operated in the embedding space instead of discrete space of the input. They saved the semantic meaning of the sentence by restricting the directions of perturbations to find a substitutional word that is in a pre-defined vocabulary instead of any unknown word. The previous two methods [17,18] are white-box attacks. On the other hand, Gao et al. [19] proposed DeepWordBug which is a black-box attack. First, they determined important words to change using scoring functions, then they created perturbations on these words causing a misclassification. To preserve the readability of the perturbed sentence, the authors used edit distance. In another work, Ebrahimi et al. [20] proposed a white-box attack called HotFlip working against character-level neural network classifiers. The authors used the gradients of the input vector to find the manipulations needed to fool the classifier. Inspired from DeepWordBug [19], in this paper, we propose a black-box attack using scoring strategies. Our attack is possible at the level of characters by changing specific characters of the malicious URL to fool the classifier or at segments level by replacing the full URL segment. The goal is to change the predicted label of the URL by introducing minimum alterations.

There also exist some studies on adversarial attacks on URL detection systems, but they were designed to introduce perturbations to the URL input features. Chen et al. [21] used a differential evolution algorithm to find the required minimum bytes to be changed to create an adversarial attack. Similarly, Shirazi et al. [22] tried to measure the number of features that should be changed to create adversarial attacks and the cost for each manipulation. On the other hand, Aleroud et al. [23] used features perturbation for adversarial attack generation using generative adversarial networks (GAN). The main difference between our study and previous studies is that

we perform attack and defense on featureless-DL systems and it works directly on the raw data while considering the URL as a sentence or text.

3 Problem Formulation

In this section, first, we provide a real-life scenario of how the attack could happen. Then, the problem of malicious URL detection is formulated as a binary classifier. After that, a brief introduction to adversarial attacks and adversarial training is provided. Lastly, we discuss typical URL parsing as it determines the restrictions on the modifications in the input URL.

3.1 Real-Life Scenario

Fig. 1 shows an example of how the proposed attack could happen in a real-life scenario.

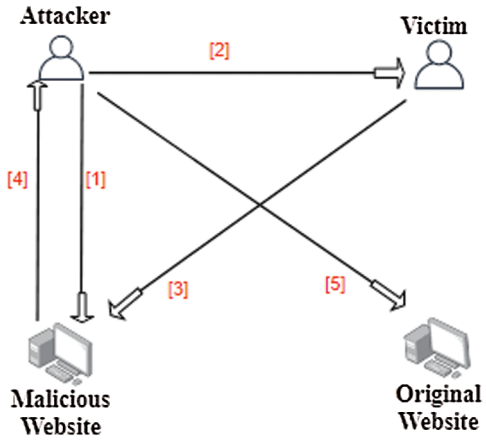

Figure 1: Real-life attack scenario

1) The attacker designs and runs a malicious website with a URL generated using our proposed attack. This website could ask the victim to download some malicious files or to fill online forms with user credentials (such as password and account details) mimicking some original service (such as bank website).
2) The attacker uses different techniques to lure or redirect the victim to the malicious website, such as social engineering, phishing, email spoofing, website spoofing, and exploitation of browser vulnerabilities. In our case, the victim could be using a machine learning-based classifier to detect malicious URLs, but since the attacker uses the proposed attack for generating the malicious URL, the classifier is not able to detect it.
3) Once the victim browses the malicious URL, the victim will be redirected to the webserver on which the attacker has hosted the malicious webpage website.
4) The victim executes what the attacker wants, and the attacker gets the information he/she needs.
5) The attacker uses the information provided by the victim to access the original website using the victim's credentials.

3.2 Malicious URL Detection

Consider a dataset consisting of N URLs with their corresponding labels, $\{(u_1, y_1), \ldots, (u_N, y_N)\}$, where $(u_i : i = 1, 2, \ldots, N) \in U$ represents a URL, U is the URL space which preserves

that the input has a URL parsing, and $y_i \in \{0, 1\}$ is the label of the URL, with $y_i = 1$ for a malicious URL, and $y_i = 0$ for a benign URL. First, we need to convert each URL to its feature representation $u_i \rightarrow x_i$ where $x_i \in R^n$ is an n-dimensional vector representing the features of the URL u_i. In the case of deep learning, this can be done automatically. Thus, all we need is a prediction function $f(u_i) : R^n \rightarrow R$ that can perform both feature learning and URL classification. This function f is represented as the neural network architecture of the classifier such as a CNN and recurrent neural network (RNN). The result of this function is the probabilities of each class being benign $P_f(ben \mid u_i)$ or malicious $P_f(mal \mid u_i)$ and the final output $\hat{y}_i$ is the maximum of the probabilities and denoted as:

$$\hat{y}_i = argmax\left(P_f(mal \mid u_i), P_f(ben \mid u_i)\right) \tag{1}$$

The goal is to learn the parameters or weights of the prediction function f that can minimize the number of prediction errors in the entire dataset. To accomplish this, we need to choose and minimize a loss function, where different loss functions can be used, such as the mean squared error and the cross entropy.

3.3 Adversarial Attacks

Adversarial attacks are security vulnerabilities in ML and DL models. Adversaries can utilize these attacks to fool DL models by altering samples with a small perturbation invisible to humans. Formally, for a given DL classifier f, a small perturbation Δx performed on an input sample x results in a new sample x' as an adversarial sample:

$$x' = x + \Delta x, \quad \|\Delta x\|_p < \varepsilon, x, x' \in X \tag{2.1}$$

$$f(x) \neq f\left(x'\right) \quad \textbf{or} \quad f\left(x'\right) = t \tag{2.2}$$

The classifier here is denoted as $f: X \rightarrow Y$, where X is the input sample space and Y is the output classes' space. $\|\Delta x\|_p$ is the $L_p - norm$ of the perturbation Δx and it measures the degree to which an adversarial example x' is imperceptible from its original x. ε is the permitted perturbation so that the x' stays in the input space and that it is indistinguishable from x by a human observer. The constraint in (2.2) means that the class of the adversarial sample can either be any other class different from the original sample in untargeted attacks or a specific one (t) in targeted attacks.

In our case, the input sample is a malicious URL u and we target finding a perturbation that changes u into u' so that u' is classified as a benign URL:

$$u' = u + \Delta u, \quad \|\Delta u\|_p < \varepsilon, u, u' \in U \tag{3.1}$$

$$f(u) = 1 \quad (malicious) \tag{3.2}$$

$$f\left(u'\right) = 0 \quad (benign) \tag{3.3}$$

The constraints in (3.1) are $\left(u' \in U\right)$ mean that u' should have a typical URL parsing style and $\left(\|\Delta u\|_p < \varepsilon\right)$ means that Δu should be as minimum as possible and less than a predefined permitted perturbation parameter ε. The constraints in (3.2) and (3.3) mean that the input sample u is classified as malicious and the perturbed sample u' should be classified as benign.

Adversarial attacks are divided into white-box and black-box attacks according to the amount of knowledge that the attacker has about the model [24]. In white-box attacks, the attacker knows

the architecture and weights of the model which makes the attack easier to launch. Whereas in black-box attacks, the attacker has no or limited knowledge about the model which is difficult but more realistic in some cases. If the attacker has no knowledge of the model such as the training process and weights but can query the model for investigating input samples, it is called adaptive black-box attacks. The attacker constructs adaptive queries to the target model by changing the queries according to the label y obtained from the target model for a sample x. The attacker then builds a surrogate model and trains it on (x, y) obtained from querying the target model [24]. This surrogate model replaces the target model and white-box attacks can be constructed on it. In this study, we consider adaptive black-box attacks as these attacks are more realistic since the model could be deployed somewhere on the Internet as a browser extension or as a spam email detector.

3.4 Adversarial Training

Since it is essentially important to mitigate the effects of adversarial attacks, many robust DL methods and models have been proposed. Some of these methods are adversarial training, defensive distillation, and using GANs as defense mechanisms against adversarial attacks [13]. It is important to mention that each of these defense methods is used to defend against specific attacks, and none of them alone can mitigate all kinds of adversarial attacks.

In this paper, we examine the effect of using adversarial training as a defense for the classifier. In [25], the authors introduced adversarial training to secure the model by augmenting the training dataset with perturbed data received from attacking the original dataset. Different adversarial attacks are used to generate the adversarial examples then augment these perturbed data to the model's training data. Formally, first, for each input sample, all samples that cause maximizing the loss function should be found using the proposed attack. Then, during the training of the model, instead of updating the parameters depending only on the loss of original samples, the pre-treated input is included as follows.

$$\delta_{i+1} = \arg\max_{\delta \in \Delta} I[f(x+\delta \mid \theta_i) \neq y] \tag{4.1}$$

$$\theta_{i+1} = \arg\min_{\theta} L(x+\delta_{i+1}, y, \theta) \tag{4.2}$$

I is an Indicator function, f is the classifier, L is the loss function, Δ is the allowable perturbation, δ is the perturbation that leads to maximizing the loss function of the classifier for an input sample x. In (4.2) this perturbed sample is included in the training process of the model to find the optimal parameters of the model θ.

3.5 URL Parsing

URLs have a specific structure (as shown in Fig. 2) that determines how to forward the request from the user to the end servers. A URL includes the following segments:

- A protocol or scheme identifier (e.g., HTTP, HTTPS),
- A Domain name or netloc registered in DNS server,
- A path which identifies where the page is located in the server,
- Query: name and value pairs as parameters provide information that the resource can use for some purpose, and
- Fragment: used to direct the browser to a reference or a function.

URL parsing breaks it down into its partial components or segments allowing it to be treated as words in a text sentence. All URL components can be replaced with any other suitable

alternative except for the domain name which is the only part the programmer cannot replace arbitrarily as it is registered in DNS servers. For example, changing the path will not change the content as long as the programmer is able to place the page on a different path. We will try to exploit this feature to build the attacks.

Figure 2: URL structure

4 Proposed Attack on Featureless URL Classifiers

We propose an adversarial attack that is possible at the level of characters by changing specific characters or at the segments level by replacing the full URL segment. In this study, we refer to characters or segments as 'token,' depending on the attacker's purpose. We also note that the previous definition for adversarial modification given in (3.1): $\Delta u = u' - u$ cannot be directly applied here because the input URL sample is symbolic and $L_p - norm$ works for continuous image data; yet, it is meaningless for text data. Therefore, replacement modifications of characters or words into the text are proposed to alter a malicious URL into a benign URL. The definition of adversarial modifications here is the edit distance between input URL u and perturbed URL u' that is defined as the minimal edit operations required to change u to u'. The goal is to change the predicted label of the URL by introducing minimum alterations. Moreover, we apply scoring strategies inspired by DeepWordBug [19] to find the important URL segments or characters that, if changed, can cause the misclassification needed for (3.1) and give an incorrect prediction as in (3.2), (3.3). The proposed attack follows a black-box scenario as it is more realistic to assume that the model is deployed somewhere as a part of a security system on cloud servers as a service. Such service may receive the input from the users and returns corresponding outputs so the architecture and parameters or gradients of the neural network used for classification are not available. Therefore, the required adversarial modifications are created for the chosen tokens of the input without considering the gradients or weights of the model. With the numerous choices of potential input changes (among all the URL segments/characters changes), we design an approach that consists of three steps to choose the tokens and replace them:

- Step 1: Determine the important malicious URL tokens to change,
- Step 2: Determine the important benign URL tokens or candidates pool, and
- Step 3: Introduce the potential attacks that can evade the deep learning classifier with more than one suggestion if possible.

To find tokens for the first two steps, we design scoring functions. However, the proposed changes should preserve the structure of the URL. The scoring functions first find the important token in the malicious input URL and potential candidates to replace them and then the attack executes a specific modification in those tokens that causes the classifier to misclassify the URL. Moreover, to find important tokens, we define scoring functions (discussed in Subsections 4.1 and 4.2), which are designed to evaluate which tokens affect the decision of the target model more in both malicious and benign cases. After detecting the tokens to be changed in the malicious URL and the best candidates to replace them from the benign URL, the modification is applied to form an adversarial sample. We start by finding the score of each input URL's segment towards determining the maliciousness of the input URL with respect to the classifier f. Next, the URL

segments are ranked according to their score in decreasing order. Thus, we ensure that the task is accomplished with the least possible number of steps. Then, we iterate through the URL segments in decreasing order of importance and transform the token to a new token. Unless the segment is a domain name, it is possible to replace it with another segment, otherwise, the domain name must be tackled by characters to create a new domain name that is not registered in the DNS server.

4.1 Determining the Important Malicious URL Tokens to Change

The ultimate goal is to find a perturbation that leads to misclassifying a malicious input URL as a benign URL. For this purpose, first, we define a scoring function $SCR_m\,(token_i) : R^n \rightarrow R$ to determine the important tokens used by the classifier for measuring the maliciousness of an input (malicious) URL. Then we calculate the importance of tokens in an input URL according to the contribution to the classification confidence or the class probabilities resulting from the classifier and after that rank the tokens in the decreasing order of their importance. Formally, the score of a malicious URL's token is calculated as follows:

$$\begin{aligned} SCR_m\,(token_i) &= \left(P_f\left(ben \mid u^{token_i}\right) - P_f\,(ben \mid u)\right) - \left(P_f\left(mal \mid u^{token_i}\right) - P_f\,(mal \mid u)\right) \\ &= \left(P_f\left(ben \mid u^{token_i}\right) - P_f\,(ben \mid u)\right) - \left(\left(1 - P_f\left(ben \mid u^{token_i}\right)\right) - \left(1 - P_f\,(\text{ben} \mid u)\right)\right) \\ &= 2 * P_f\left(ben \mid u^{token_i}\right) - 2 * P_f\,(ben \mid u) \end{aligned} \tag{5}$$

To calculate the importance of a token, we measure the effect of this token on the probability of this URL belonging to a benign or malicious class. This is accomplished by subtracting the difference in prediction probability for the benign class and the malicious class before and after deleting the token. In the second line of (5), we use the fact that the sum of the two output probabilities (the probability for an input URL to be malicious or benign) is equal to one since we use a binary classifier. Large score values mean that deleting this token would lead to a larger benign-ness probability or less maliciousness probability. Hence, exchanging this value with another value can lead to a higher benign-ness probability.

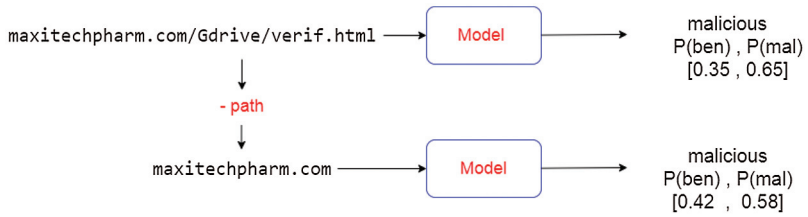

Figure 3: An example of calculating the score of the path in a given URL

Perturbing tokens that have high scores leads to less maliciousness and higher benignness scores. Finding the scores can be achieved at the segment's level for all URL segments except for the domain name where the score is calculated at the character's level. This is due to the attacker's ability to replace any segment with another segment inside the site; however, for the domain name, the attacker should create a new unregistered domain name. As shown in Fig. 3, the score of the

path in the given URL is positive and has a relatively high value; therefore, replacing this path with another one from a benign URL could increase the benignness probability of this URL. Once we estimate the importance of each token in the input sequence, they can be ranked and the top n tokens are selected for perturbation to create an adversarial sequence, where n is the number of allowed perturbations.

4.2 Determining the Important Benign URL Tokens

Here the main goal is to build a benign candidate pool for each token in the malicious URL. The candidate pool for each token consists of the scored benign tokens that can replace this token. The scoring function in this step finds the important tokens used by the classifier to determine the benignness of a URL. Formally, the score of a benign URL's token is calculated using the following formula:

$$\begin{aligned} SCR_b(token_i) &= \left(P_f\left(mal \mid u^{token_i}\right) - P_f(mal \mid u)\right) - \left(P_f\left(ben \mid u^{token_i}\right) - P_f(ben \mid u)\right) \\ &= \left(P_f\left(mal \mid u^{token_i}\right) - P_f(mal \mid u)\right) - \left(\left(1 - P_f\left(mal \mid u^{token_i}\right)\right) - \left(1 - P_f(mal \mid u)\right)\right) \\ &= 2 * P_f\left(mal \mid u^{token_i}\right) - 2 * P_f(mal \mid u) \end{aligned} \tag{6}$$

The concept is the same as in the previous step, i.e., tokens that have high scores are chosen, and changing them leads to higher maliciousness and less benignness. We start by calculating the scores of all tokens in all benign URLs in the dataset, then rank the tokens in each URL segment according to their score in descending order. Again, the calculation is at the segment level for all URL tokens except for the domain name where the calculation is at the character level.

4.3 Token Transformer

To this end, we have sorted tokens of the malicious URL from a scoring function in Step 1 (Section 4.1.) and have built the candidate pool by choosing the n important candidates for each token in Step 2 (Section 4.2.). The next part of creating the adversarial sample is to transfer or modify the tokens. The modification of a token can be done directly by replacing tokens from Step 1 with corresponding candidates from Step 2 recursively until the URL is predicted as benign by the model. This simple mechanic for perturbing malicious URLs is summarized in Algorithm 1.

Algorithm 1: Token transformation

Input: Sample URL—u, segment to change—s, number of allowed characters to change—m (default = $length(s)$)
Output: A benign URL *u_new*

```
Find a candidate pool CP for s using Step 2
Initialize ben_u = P_f(ben | u))
If s != domain_name:
    for cand in CP:
        u_new = Replace s in u with cand
        ben_new = P_f(ben | u_new))
        If ben_new > ben_u:
            ben_u = ben_new
```

(Continued)

```
            u = u_new
Elif s = domain_name:
    for i in range(m):
        for j in range(length(s)):
            for cand in CP:
                u_new = Replace [j: j+i] portion in the domain name of u with cand[j: j+i]
                ben_new = P_f (ben | u_new))
                If ben_new > ben_u:
                    ben_u = ben_new
                    u = u_new
Return u
```

Since tokens are considered in the order of their contribution score SCR ($token_i$), constructing adversarial samples is achievable with the least possible changes using the greedy method. For domain name, the attack is character-based because new domain names must be created that are not registered in the DNS server. We replace a portion of the input URL with its correspondent in the candidate pool with maximum length m. The attack starts with the most important character from the hostname and replace it with its corresponding candidate and recursively replacing until it reaches an adversarial sample. If the number of characters is small and not enough to construct an attack, new characters are added to the hostname at the end. An example of characters-based domain name attack is shown in Fig. 4.

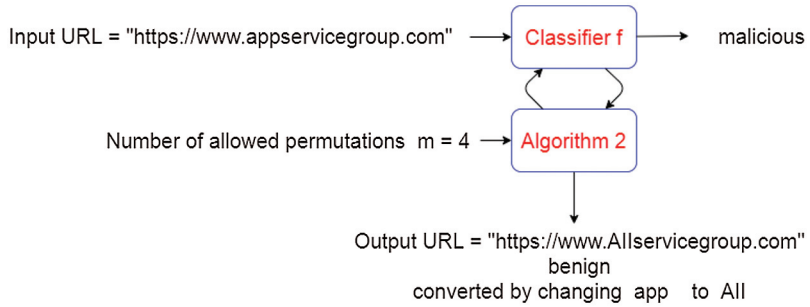

Figure 4: An example of changing a malicious input URL to a benign URL by changing three characters using the proposed attack

For other segments of the URL, both the character and segment level of the attack are available as long as the crafted URL has the standard URL parsing. We chose not to change the protocol (or the scheme identifier) if it is http or https; hence, the changes are executed on all other segments. The protocol affects the decision of the classifier only if the dataset implicates this. For example, if all benign URLs in the dataset start with https and all malicious URLs start with http, then any URL starts with http will be classified as a malicious URL with high probability. Below is the algorithm of the full proposed method for converting an input malicious URL u into a benign URL.

Algorithm 2: Convert the input malicious URL u into a benign URL

Input: Input URL to change—u, Classifier trained for malicious URL detection—f, number of allowed permutations—m (default = $length\,(parse\,(S))$)
Output: An array *Malicous_url* of malicious URLs *u_new*

Find the score $SCR_m\,(s_i)$ of each URL's segment s_i with respect to the classifier f.
Rank segments S of u according to $SCR_m\,(S) : S \rightarrow \{s_1, s_2, \ldots, s_k\}$ where, $SCR_m\,(s_1) > SCR_m\,(s_2) > \cdots > SCR_m\,(s_k)$ and k: $length\,(parse\,(S))$ number of segments in S
$i \leftarrow 1$, $y \leftarrow f\,(s)$, *Malicous_url* = [ ]
while $i <= m$ do
    if s_i is not a domain name
        u_new = Transform(u, s_i)
    else
        u_new = Transform_char(u, s_i)
    end if
    $y \leftarrow f\,(u\_new)$
    if $y = benign$:
        Malicous_url.append(*u_new*)
    $i \leftarrow i + 1$
end while

5 Experiments and Results

In this section, we discuss the implementation details and the obtained results of testing different versions of our proposed attack against three kinds of DL-based URL classifiers. Moreover, we evaluate the performance of the proposed attack by testing the accuracy of the model while increasing the number of allowed permutations. Finally, we examine adversarial training for designing more robust models against such attacks. We test our attack on three kinds of DL classifiers: (i) character-based classifier (ii) word-based classifier, and (iii) full character-based and word-based classifier. To train the DL binary classifier, we need to build a dataset containing labeled benign and malicious URLs. There are many open-source datasets for this purpose. We consider a dataset that was created to address the problem of malicious URLs classification on the Internet [26]. The dataset contains malicious and benign URLs that can be used for analysis or building classifiers. The dataset was acquired from various sources, such as PhishTank [27], etc. and it contains 450176 unique URLs of which 77% are benign and 23% are malicious.

As was mentioned earlier, in the DL approach, feature engineering is not required before feeding the URL to the model. Nevertheless, some pre-processing of the raw URLs is still needed. The characters and words of the URL need to be expressed as integers. This requires building a dictionary containing all possible characters or words in the URL. The featureless DL approach considers the problem of malicious URL classification as a text classification problem and uses techniques from NLP to solve it. After preprocessing the raw URL, an embedding layer is used to move characters or words that occur in the same context closer to each other in an n-dimensional space. This layer will be used as part of the model where the embedding is learned along with the model training process.

For the character-based classifier [28], each character is considered as a word and the position of a character within the n-dimensional vector space is learned from the URL. This position is based on the characters that surround this character as a context as shown in Fig. 5 for example.

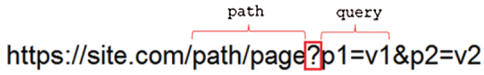

Figure 5: The character '?' indicates the end of the path segment and the beginning of the query segment

The vocabulary size, which contains allowed characters for embedding, here is 100 and it contains all unique printable characters in python. As a result, each character has its own embedded vector. Character-based classifiers decide the class of a URL depending on a specific set of characters appearing together. On the other hand, for word-based classification, all unique words in the training dataset are considered. Because there are no restrictions on the words in the URL, and especially in the domain name, the vocabulary size here extends unlimitedly as the training dataset gets bigger. To accomplish that, we replace words that appear less than two times with "unknown." Word-based classifiers determine the class of a URL depending on a specific set of words appearing together.

For a complex word-level and character-level classifier, we maintain the same embedding concept proposed by [10]. In this embedding, two matrices are used, one for words and the other for characters, whereas the final URL representation is the sum of these two matrices. The input URL u to a classifier is converted into a 2D matrix: $u \rightarrow x \in R^{L*K}$ where L is the maximum length of the component that can be a word or a character for each URL and K is the embedding dimension. There are several architectures to build the model for this task, for instance, LSTM architecture and 1D convolutions. To this end, we use CNNs as they can discover important information for the classification from groups of characters or words appearing together in the URL which could indicate if a URL is benign or malicious [29,30]. In 1D convolutions, the width of the sliding window is constant which, in this case, is the embedding dimension K. The convolution process happens over $x \in R^{L*K}$. We use the same CNN classifier architecture proposed by [10] by using 4 convolutional filters $W \in R^{K*h}$, where h = 3, 4, 5, 6 and for each filter size, 456 filters are used. Using these filters, the network can consider the relationship among h components (characters or words) appearing together. In this study, we consider cross entropy as a loss function for training all classifiers.

5.1 Performance Evaluation

To measure the performance of the proposed attacking methods, the accuracy of the model is observed on the generated adversarial samples. Effective attacks lead to lower accuracy as they are able to successfully fool the classifier. We test three variants of the attack as follows: (i) segment-based attack for all URL segments except for the domain name, (ii) character-based attack for the domain name, and (iii) full attack where both character-based and segment-based attacks are used to create the final attack. To study the influence of the dataset size on the accuracy and the robustness of the model, we conduct experiments on the full dataset and partial dataset by taking 100k training samples with preserving the basic proportions between benign and malicious samples. We also measure the effectiveness of each of these attacks by determining the reduction in the correct classification rates of malicious URLs, i.e., first, adversarial samples are constructed, then the percentage of those that are correctly classified as malicious URLs is found. The results of these experiments are presented in Tab. 1.

From Tab. 1, we can see that the best base-ground accuracy is achieved by applying the model proposed in [10] which uses both characters and words to classify the sample. We also note that

the proposed attack was able to reduce the accuracy of all models. Character-based classifiers have better accuracy and robustness than word-based classifiers. That is due to the large vocabulary size in these applications and the freedom that the programmer has to name the site and the path inside this site. We also observe that, when the number of allowed characters to change equals to five, the full attack achieved a 56.3% reduction in the accuracy of the full character+word-level classifier, 60.1% of the character-based classifier, and 67.5% of the word-based classifier. It is worth mentioning that the attack works better for small datasets; however, it achieved a 56.3% reduction in the accuracy of the best model making the use of these models insufficient in real protection systems where attackers try every possible way to attack the system.

Table 1: Reduction in the accuracy of three classifiers against the proposed attack

	Classifier with training-set ≈ 500K			Classifier with training-set ≈ 100K		
	Character-level	Word-level	Full character + word-level	Character-level	Word-level	Full character + word-level
Base-ground accuracy	94.1	92.7	95.6	90.3	87.9	91.1
Segment-based attack	76.9	69.3	80.7	72.8	65.5	77.1
Character-based attack	53.9	43.6	56.1	43.3	32.4	47.5
Full attack	34.0	25.2	39.3	22.5	16.1	30.8

For the full attack, we observe the effectiveness of the attack by increasing the number of allowed characters to change in the domain name. As shown in Fig. 6, the accuracy of the model starts to drop significantly when the number of characters allowed to change is more than two.

Moreover, to measure the effectiveness of our scoring functions, the proposed attack is compared with the following four random-based attacks that depend on introducing random changes on the URL:

1) Attack 1: Remove the tokens randomly from the malicious URL without replacing them with candidates from benign URLs. Four characters are allowed to be deleted from the domain name and all other segments to be completely deleted.
2) Attack 2: Order the tokens from the malicious URL according to their cost and pick tokens randomly from benign URLs for replacement.
3) Attack 3: Order the tokens randomly from the malicious URL and replace them with randomly selected tokens from benign URLs.
4) Attack 4: Order the tokens randomly from the malicious URL and replace them with tokens from benign URLs ordered according to their cost.

The reduction in the correctly classified malicious URLs rates for the three previous classifiers against these random-based attacks is shown in Tab. 2. The results show that random-based attacks influence all classifiers considerably less than our proposed attack; thus, the proposed ordering and replacing approach using scoring functions is important. Similarly, the arrangement of the classifiers according to their robustness against random attacks is the same as against our proposed attack. We also note that attack 4 influences the classifiers more than other attacks because it orders benign candidates according to their cost. Next, attack 2 takes place because it orders malicious tokens according to their cost.

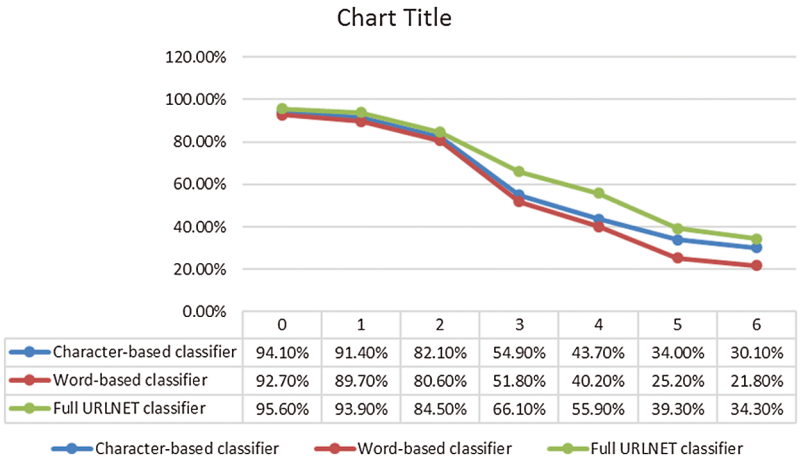

Figure 6: Change in model accuracy according to the number of changed characters in the domain name

Table 2: Reduction in the accuracy of three classifiers against the random-based attacks

	Classifier with training-set $\approx$ 500K		
	Character-level	Word-level	Full character + word-level
Attack_1	80.51	79.11	83.50
Attack_2	74.80	72.86	75.07
Attack_3	78.20	77.04	79.269
Attack_4	68.66	67.13	70.75

5.2 Increasing the Robustness with Adversarial Training

Next, we study the impact of adversarial training on mitigating the effect of small perturbations on input samples. In an attempt to understand how adversarial training could increase the robustness of the classifier, the above-mentioned experiments are repeated to observe the difference in the performance between standard and adversarially trained model using two training datasets with different sizes. However, we restrict ourselves to augmenting the full classifier (as it achieved the best results), in an effort to make it more robust. The goal of using adversarial training is to improve the generalization performance of the classifier by considering crafted samples outside the original training dataset. An effective generalization reduces the classifier sensitivity to minor perturbations which increases its resilience against adversarial attacks.

Applying adversarial training is done by performing the attacks on all malicious URLs and then the perturbed URLs are augmented to the training set as malicious URLs (as shown in Fig. 7). Using the same approach suggested earlier, the number of characters allowed to change in the domain name (m) ranges from one to six. Thus, we fix (m), find all adversarial samples, augment the samples to the training dataset, train the model again, and finally, we try the attack on the final model and compare the accuracy before and after applying adversarial training. The results are shown in Tab. 3.

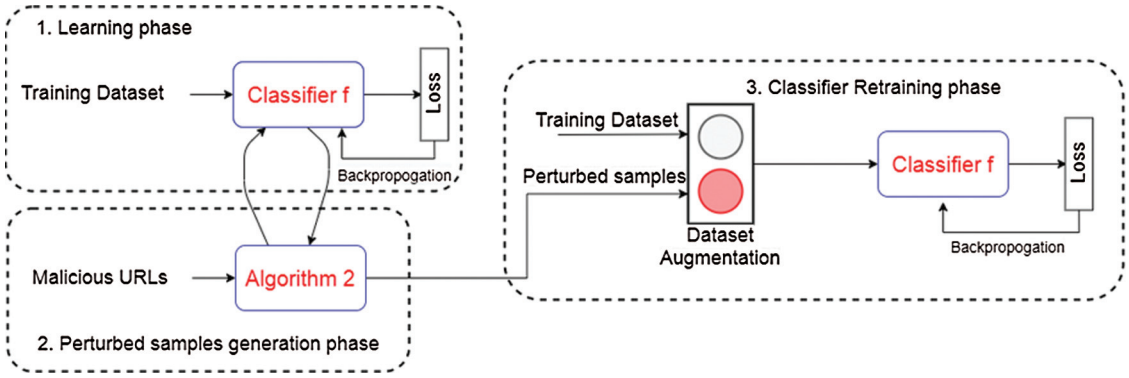

Figure 7: Applying adversarial training by retraining the model on dataset received from augmenting the original dataset with adversarial examples received from applying the proposed attack

Table 3: Increase in the rate of correctly classified malicious URLs for three classifiers against the proposed attack after applying adversarial training

	Classifier with training-set ≈ 500K			Classifier with training-set ≈ 100K		
	Character-level	Word-level	Full character + word-level	Character-level	Word-level	Full character + word-level
Base-ground accuracy	94.1	92.7	95.6	90.3	87.9	91.1
Segment-based attack	92.2	89.4	94.1	82.1	80.0	84.9
Character-based attack	89.4	82.8	91.7	77.1	72.8	81.2
Full attack	88.0	80.9	90.4	76.5	73.1	80.7

The columns in Tab. 3 represent the attack that is used to augment the training dataset for adversarial training and the rows represent which attack is tried after re-training. Adversarial training shows promising results in reducing the influence of the attack on the robustness of the model. The influence of the segment-based attack was reduced to less than 3% on average. The influence of the character-based attack and the full attacks was reduced to less than 7% on average. Furthermore, it was observed that a larger training dataset size leads to a smaller gap between the accuracies of standard and adversarial training models.

6 Discussion

We aim at estimating the robustness of DL-based malicious URL classification systems under adversarial attacks. The results of this study show that these systems are not applicable until the robustness of the model is considered and not just the accuracy. By observing the results, one of the challenges that can make these systems less secure than other NLP DL-based systems is the vast number of unique words that appear in the URL which do not exist in the formal English language. This is due to the fact that the domain name and other variables are set by the programmer without any rules of how to name these variables. Using n-gram could reduce the effect of this problem by sliding a window of n characters over the URL domain name to generate n-gram tokens. This solution could alleviate the problem but does not solve it completely when the attacker adds n-gram from a benign URL. Another problem with word-based classifiers is that the attacker can replace all segments (except for the domain name) of the malicious URL with

segments from a benign URL, which could reduce the maliciousness of the URL significantly. To address this issue, the domain name of any full URL classified as malicious should be added to a blacklist, thus changing the path and other segments of the URL would not result in classifying it as benign. Here are some recommendations to make these systems more robust against these kinds of attacks:

1) Include the pre-treated input in the loss function which reduces the model's sensitivity to small changes.
2) Consider a large training dataset with a reasonable amount of malicious URLs which increases the accuracy and robustness of the model.
3) Create some predictive features such as the length of the URL, the existence of executable extensions in the URL, the existence of redirection in the URL, etc. Although these features require domain knowledge and increase the complexity of the model, they harden the process of crafting adversarial samples. Add the domain name to a blacklist after being discovered with this model to prevent changing other segments which leads to misclassification from happening.

7 Limitations

Our study has some potential limitations. For instance, we performed the experiments on an open-source, publically available dataset; however, getting a larger dataset with more up-to-date malicious URLs could lead to more precise results. Moreover, the proposed attack was tested against classifiers that we designed ourselves. Testing the attack against real classifiers used by security companies and web browsers would be more realistic. However, security companies usually do not share the architecture of their malicious URL classifiers that is why we could not test it against those classifiers. Besides, testing against these classifiers is considered illegal. Furthermore, their classifiers are usually a mixture of various approaches such as machine learning methods, blacklist-based techniques, heuristic approaches, etc. For adequate security, attacks on each employed strategy and their defense mechanisms should be separately studied. Nevertheless, our proposed attack highlights the security issues regarding the potential use of DL-based featureless methods for malicious URL detection in industrial solutions.

8 Conclusions

In this paper, we investigated the robustness of featureless malicious URLs detection models and proposed a black-box attack against these models. This attack exploits the sensitivity of NLP-based classifiers against small purposely crafted perturbations. The attack can work at segment level, character level, or use both segments and characters changes to fool the classifier. All changes should preserve the typical URL parsing. In essence, we examined three CNN-based classifiers: Character-based, word-based, and jointly word and character. The results of our experiments show that the attack causes a 56% decrease in the classification accuracy for a joint model, a 77% decrease for a word-level model, and a 60% decrease for a character-level model. Furthermore, we also used adversarial training to increase the robustness of the model converting this attack in order to augment the training data. Lastly, we introduced some recommendations that should be considered when designing such systems. The results of this paper indicate that there are still loose ends to further study before applying such systems in real-life security applications and this is related to the progress in defending against adversarial attacks on deep learning models.

Funding Statement: This research was supported by Korea Electric Power Corporation (Grant Number: R18XA02).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] G. Varshney, M. Misra and P. K. Atrey, "A survey and classification of web phishing detection schemes," *Security and Communication Networks*, vol. 9, no. 18, pp. 6266–6284, 2016.

[2] AWPG, "Phishing activity trends report 1st quarter," *Most*, 2020. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2020.pdf.

[3] J. Ma, L. K. Saul, S. Savage and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Paris, France, pp. 1245–1253, 2009.

[4] M. Bortnikov, A. Khan, A. M. Khattak and M. Ahmad, "Accident recognition via 3D CNNs for automated traffic monitoring in smart cities," in *Science and Information Conf.*, Las Vegas, NV, USA, vol. 944, pp. 256–264, 2019.

[5] B. J. Akinsanya, L. J. P. Araújo, M. Charikova, S. Gimaeva, A. Grichshenko *et al.,* "Machine learning and value generation in software development: A survey," *Software Testing, Machine Learning and Complex Process Analysis*, vol. 1, pp. 1–10, 2019.

[6] F. Hussain, R. Hussain, S. A. Hassan and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 1686–1721, 2020.

[7] S. Seufert and D. O'brien, "Machine learning for automatic defence against distributed denial of service attacks," in *IEEE Int. Conf. on Communications*, Glasgow, UK, pp. 1217–1222, 2007.

[8] A. Keliris, H. Salehghaffari, B. Cairl, P. Krishnamurthy, M. Maniatakos *et al.,* "Machine learning-based defense against process—Aware attacks on Industrial Control Systems," in *IEEE Int. Test Conf.*, Fort Worth, TX, USA, pp. 1–10, 2016.

[9] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li *et al.,* "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[10] H. Le, Q. Pham, D. Sahoo and S. C. H. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," *arXiv*, no. i, 2018. [Online]. Available: http://arxiv.org/abs/1802.03162 [Accessed: May 10, 2020].

[11] M. Kührer, C. Rossow and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8688. Cham: Springer, pp. 1–21, 2014.

[12] I. Namatēvs, "Deep convolutional neural networks: Structure, feature extraction and training," *Information Technology and Management Science*, vol. 20, no. 1, pp. 40–47, 2018.

[13] S. Qiu, Q. Liu, S. Zhou and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, pp. 909, 2019.

[14] J. Saxe and K. Berlin, "eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys," 2017. [Online]. Available: http://arxiv.org/abs/1702.08568 [Accessed: May 10, 2020].

[15] K. Shima, D. Miyamoto, H. Abe, T. Ishihara, K. Okada *et al.,* "Classification of URL bitstreams using bag of bytes," in *21st Conf. on Innovation in Clouds, Internet and Networks*, Paris, France, pp. 1–5, 2018.

[16] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[17] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," 2017. [Online]. Available: http://arxiv.org/abs/1707.02812 [Accessed: May 10, 2020].

[18] M. Sato, J. Suzuki, H. Shindo and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," in *Int. Joint Conf. on Artificial Intelligence*, Stockholm, Sweden, vol. 2018, pp. 4323–4330, 2018.

[19] J. Gao, J. Lanchantin, M. L. Soffa and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *Proc.—2018 IEEE Symp. on Security and Privacy Workshops*, San Francisco, CA, USA, pp. 50–56, 2018.

[20] J. Ebrahimi, A. Rao, D. Lowd and D. Dou, "Hotflip: White-box adversarial examples for text classification," in *ACL 2018—56th Annual Meeting of the Association for Computational Linguistics, Proc. of the Conf. (Long Papers)*, Melbourne, Australia, vol. 2, pp. 31–36, 2018.

[21] W. Chen, Y. Zeng and M. Qiu, "Using adversarial examples to bypass deep learning based URL detection system," in *Proc.—4th IEEE Int. Conf. on Smart Cloud, Smart Cloud 2019 and 3rd Int. Symp. on Reinforcement Learning*, Tokyo, Japan, pp. 128–130, 2019.

[22] H. Shirazi, B. Bezawada, I. Ray and C. Anderson, "Adversarial sampling attacks against phishing detection," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Cham: Springer, vol. 11559, pp. 83–101, 2019.

[23] A. AlEroud and G. Karabatis, "Bypassing detection of url-based phishing attacks using generative adversarial deep neural networks," in *Proc. of the Sixth Int. Workshop on Security and Privacy Analytics*, New Orleans, LA, USA, pp. 53–60, 2020.

[24] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay and D. Mukhopadhyay, "Adversarial attacks and defences: A Survey," 2018. [Online]. Available: http://arxiv.org/abs/1810.00069 [Accessed: May 12, 2020].

[25] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and harnessing adversarial examples," 2015. [Online]. Available: http://arxiv.org/abs/1412.6572 [Accessed May 22, 2020].

[26] "Malicious and Benign Websites | Kaggle," 2019. [Online]. Available: https://www.kaggle.com/siddharthkumar-25/malicious-and-benign-urls [Accessed May 13, 2020].

[27] LLC OpenDNS, "PhishTank: An anti-phishing site," 2016. [Online]. Available: https://www. phishtank.

[28] X. Zhang, J. Zhao and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in Neural Information Processing Systems*, vol. 1, pp. 649–657, 2015.

[29] C. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proc. of COLING 2014, the 25th Int. Conf. on Computational Linguistics: Technical Papers*, Dublin, Ireland, pp. 69–78, 2014.

[30] Y. Kim, "Convolutional neural networks for sentence classification," in *Conf. on Empirical Methods in Natural Language Processing, Proc. of the Conf.*, Doha, Qatar, pp. 1746–1751, 2014.