Robust Phishing Detection Against Adversaries

SAAD AL-AHMADI
Computer Science Department
King Saud University
PO Box 51178, Riyadh 11543
SAUDI ARABIA

Abstract: - Phishing websites have grown more recently than ever, and they become more intelligent, even against well-designed phishing detection techniques. Formerly, we have proposed in the literature a state-of-the-art URL-exclusive phishing detection solution based on Convolutional Neural Network (CNN) model, which we referred as PUCNN model. Phishing detection is adversarial as the phisher may attempt to avoid the detection. This adversarial nature makes standard evaluations less useful in predicting model performance in such adversarial situations. We aim to improve PUCNN by addressing the adversarial nature of phishing detection with a restricted adversarial scenario, as PUCNN has shown that an unrestricted attacker dominates. To evaluate this adversarial scenario, we present a parameterized text-based mutation strategy used for generating adversarial samples. These parameters tune the attacker's restrictions. We have focused on text-based mutation due to our focus on URL-exclusive models. The PUCNN model generally showed robustness and performed well when the parameters were low, which indicates a more restricted attacker.

Key-Words: - Artificial intelligence, Convolutional Neural Network (CNN), Cyber Security, Phishing Detection

Received: March 7, 2021. Revised: October 5, 2021. Accepted: December 8, 2021. Published: January 14, 2022.

1 Introduction

Phishing is defined as "a scalable act of deception whereby impersonation is used to obtain information from a target" [1]. Phishing can be performed through various means such as SMS, phone calls, emails, and websites. We are explicitly considering phishing websites. Phishing websites are attempts to impersonate other websites or entities for various reasons, such as convincing users to enter their personal information. Phishing websites can be dangerous, especially for amateur users.

There are various approaches in the literature for protecting against phishing websites. One of which is employing machine learning for the automatic detection of phishing websites. However, we note that phishing detection is adversarial in nature, unlike many machine-learning tasks. In some situations, the attacker can attempt to avoid detection by mutating an existing phishing instance or creating a new phishing instance. Famous machine learning evaluation metrics do not show how the models would perform in these adversarial situations.

In this paper, we are interested in evaluations under adversarial situations. For the evaluations, we propose the following simple adversarial scenario. We assume the targets of phishing are in protected networks, which may block the attacker or issue a special warning if there are many detected phishing attempts. This scenario effectively gives the attacker a limited number of attempts. We assume that the attacker wants to use an already existing phishing instance and wants to make a small change in the phishing instance automatically to bypass the detection. The attacker's motivation is that the modified phishing instance should be very close to the existing phishing instance so that the user can still fall for it. We choose this adversarial scenario because it restricts the attacker, not necessarily reflecting a common situation in practice. In less restricted adversarial scenarios, it is easy for the attacker to bypass detection, as seen in section 2. Fig.1 illustrates the proposed adversarial scenario.

Additionally, we are specifically interested in URL-exclusive models which depend on the URL characters only. These models do not depend on any third-party service or network connectivity. This independence is useful in various situations, such as with firewalls, which may have limited storage, restricted connectivity, and need for high throughput, which may make more complex approaches unfeasible. We note that URLs that are textual in nature bring new challenges to simulating

the adversarial scenario in comparison to numerical features and other simpler features. In section 2, we will discuss a simulating strategy which is not applicable to textual features.

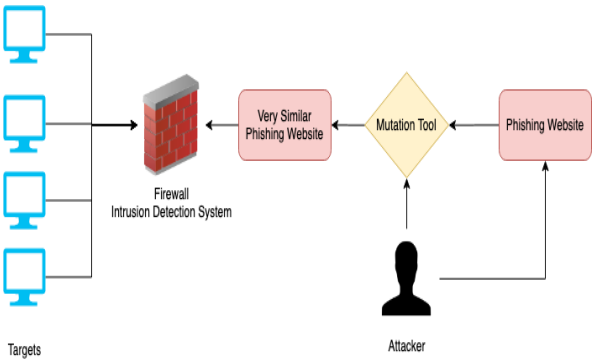

Fig. 1: Illustration of our Adversarial Scenario

To simulate the adversarial scenario, we propose a mutation strategy that works on the URL text with two parameters to simulate the attacker's adversarial behavior. These parameters tune the attacker's restrictions. Whereas smaller parameters indicate a more restricted attacker. We use the mutation strategy to evaluate PUCNN [2].

PUCNN is a state-of-the-art URL-only phishing detection model based on a character level Convolutional Neural Network (CNN) [3]. We applied the proposed mutation strategy to the phishing instances in the testing dataset from [2]. Finally, we report the relevant performance metric, recall (2).

2 Related Work

We mainly focus on discussing and evaluating works that consider adversarial settings. In addition, we discuss relevant CNN models which are applicable to the problem of phishing URLs detection.

Biggioa et al. [4] discussed three attacker goals in adversarial machine learning. They are security violation, attack specificity, and error specificity. In the security violation, the attacker's goal is to compromise security metrics such as availability, privacy, and integrity. To compromise the availability, the attacker may use denial of service attacks. The attacker may seek to obtain sensitive information from the model by reverse engineering, thus compromising privacy. The attacker may seek to compromise integrity, for example, by compromising the accuracy of the model. For the attack specificity, the attacker seeks for the model to misclassify specific types of instances (such as phishing). In the error specificity, the attacker seeks to increase a specific type of error. All these goals also can apply to phishing detection. In this paper, we are mainly interested in goals that affect the accuracy of the model, such as attack specificity and error specificity.

Adebowale et al. [5] used two combined deep-learning techniques, convolutional neural network (CNN) and long short-term memory (LSTM) to build a hybrid classification model named Intelligent Phishing Detection System (IPDS). The model targeted URLs and websites content such as images, text, and frames, unlike our URL-based model. The CNN+LSTM classifier was trained by using one million phishing and legitimate URLs and over 10,000 images. The proposed IPDS achieved a classification accuracy of 93.28%. They concluded that combining CNN with LSTM led CNN architecture to better result in terms of accuracy and shorter training time.

Shirazi et al. [6] investigated the robustness of machine learning-based phishing detection solutions in adversarial settings. In the investigations, they concluded that machine learning phishing detection is susceptible to adversarial learning techniques, on which attackers attempt to fool the classifier through manipulated input. They proposed simulating attacks by generating adversarial samples using direct feature manipulation on phishing instances. The authors managed to drop the recall of well-known classifiers to 70% by manipulating a single feature. Furthermore, by manipulating the four features, they have managed to drop the recall to 0%.

To perform the experiments, Shirazi et al. had to specify their threat model. In their threat model, they assumed that the attacker's goal is to attack the recall of the machine learning model by getting the adversarial samples classified as legitimate. They have assumed the attacker knows the model type and the feature set but does not know the model, its training parameters, and the datasets used. Additionally, they have assumed that the attacker has unlimited access to the model's prediction function, meaning that the attacker can generate many adversarial samples testing them against the model. Furthermore, they have ruled out poisoning attacks, where the attacker can add malicious samples to the training dataset.

Additionally, they have assumed that the attacker has full control of the URL and the page content, except that the attacker cannot change the domain in the URL. Excluding the domain make domain-exclusive models unaffected by the adversarial simulating. In this paper, we will

investigate the usage of the domain-exclusive model. For generating an adversarial sample, a phishing instance's values can be modified to any value that has already appeared in another phishing instance. This approach is not applicable to textual features such as the URL. For phishing instances that are correctly classified, all possible adversarial samples are generated. The authors computed the adversary cost with a tuple of two values: the number of features modified and the Euclidean distance between the phishing instance and the adversarial sample.

The experiments by Shirazi et al. were on four published phishing datasets. The first dataset includes 1000 legitimate websites instances from Alexa [7] and 1200 phishing instances from PhishTank [8]. The dataset contained eight features which are: domain length, presence of a non-alphabetic character in the domain name, the ratio of hyperlinks referring to the domain name, the presence of HTTPS protocol, matching domain name with copyright logo, and matching domain name with the page title. The second dataset is by Rami et al. [9]. The dataset includes 4898 legitimate instances from Alexa and 6158 phishing instances from PhishTank. The dataset includes 30 features, which are from five categories: URL-based, abnormal-based, HTML-based, JavaScript-based, and domain-based features. The third dataset is by Abdelhamid et al. [10], which contains 1350 instances and 16 features. The fourth dataset is by Tan et al. [11], which had 5000 instances for legitimate and phishing, collected from Alexa and PhishTank. The dataset included 48 features that were extracted from the URL and HTML.

In our previous publication [2], we have proposed PUCNN, a URL-only phishing model that is based on a character level CNN model. For training and evaluation, we have collected and preprocessed MUPD (Massive URL Phishing Detection) dataset which contained 1,167,201 phishing URLs and 1,140,599 legitimate URLs. The source of phishing URLs was PhishTank, whereas the legitimate URLs were collected from DomCop top 10 million domains [12]. We have split MUPD dataset into training, validation, and testing datasets of the following proportions: 0.6, 0.2, and 0.2. PUCNN achieved 95.78% accuracy in the testing dataset. PUCNN outperformed RandomForestNLP [13], a state-of-art URL-only model, in their published dataset.

Wang et al. [14] proposed PDRCNN, a URL-only phishing detection model which is also based on a character level CNN. They trained and evaluated their model on a dataset they collected 245,385 phishing URLs from PhishTank and 245,023 legitimate URLs from Alexa top 1 million. Their model achieves 95.61% accuracy using 10-fold cross-validation. However, the authors used only CANTINA+ [15] as their only benchmark. The main problem is that CANTINA+ is not a state-of-art model. Additionally, CANTINA+ is a non-URL exclusive model. Although, the authors retrieved old phishing pages to train and evaluate CANTINA+, we believe that it is likely that many of these pages no longer represent the phishing pages as they were reported a long time before collection, which makes the benchmark less useful. PDRCNN is similarly a textual URL model and can be evaluated under the proposed adversarial scenario.

Furthermore, there exist many character-level CNN architectures in the literature, such as those in [16] and [17]. These CNN models have already achieved excellent results in various text classification and language modeling tasks. Similarly, these CNN models can be applied to the problem of phishing URL detection, and it is possible to evaluate them under the proposed adversarial scenario.

3 Methodology

In this section, we discuss the threat model and how we simulate the attacker's behavior by adversarial sampling.

3.1 Threat Model

In this subsection, we discuss the attacker's goal, knowledge, influence, control, and constraints. It is important to specify the threat model, as we attempt to simulate it. Additionally, we show how and why our threat model differs from the threat model used by Shirazi et al. [6], which we discussed in related work. Table 1 provides a comparison summary. Our main motivation for having a different threat model is that the attacker in Shirazi et al. threat model is powerful, which can be seen from the results where they found that they can reduce the recall to 0 by controlling only four attributes. This threat model reflects the adversarial scenario we discussed in section 1.

3.1.1 Attacker's Goal

In our threat model, we assume that the attacker wants to attack the recall of the model. The attacker seeks the generated adversarial samples to pass as legitimate while they are phishing. In practice, achieving this goal means that the attacker manages to send the phishing website to the user, avoiding

the model's detection. This goal can also be formulated as an error specificity goal, where the attacker wants to decrease the true positive rate. This goal is the same as Shirazi et al. attacker's goal.

3.1.2 Attacker's Knowledge

Like Shirazi et al., we assume that the attacker knows about the selected features. In contrast, we assume that the attacker does not know about the model and does not care about it because we do not want the attacker to utilize any properties of the model, which may complicate our analysis.

3.1.3 Attacker's Influence

We assume that the attacker does not have access to the training phase, which rules out poisoning attacks. However, the attacker can manipulate the features of phishing websites to avoid them being labeled as phishing. We assume that the attacker has a fixed number of usages of the predict function. Unlike Shirazi et al., we had to limit the usage of the predict function, thus increasing the restrictions on the attacker because even with these restrictions, the attacker is powerful as can be seen from Shirazi et al.'s results. Besides, having unlimited access is troubling as the computational complexity is what limits the attacker. Nonetheless, a behavior of the attacker having limited access is quite common, for example, when an attacker attempts to phish employees in a company with a custom phishing detection model, the attacker may need to limit their attempts to avoid being detected.

3.1.4 Attacker's Control

We assume that the attacker can control all features. In contrast, Shirazi et al. assumed that the domain is excluded from attacker change, which means that models that depend on the domain are not affected by the attacker at all. Accurate models that depend exclusively on the domain name are successful under Shirazi et al. evaluation. In fact, in subsection 4.3, we show a very accurate model that depends on the domains only.

3.1.5 Attacker's Constraints

To make our threat model more realistic, we also include the concept of the constraints that the attacker needs to uphold. Our primary constraint is that we assume that the attacker can only change the phishing instance slightly. This constraint is reflective of what happens in practice, although the attacker can change the phishing instances, he cannot perform all changes freely. For example, it would be very hard for an attacker to increase his phishing website's ranking. Although changing the domain to any non-taken domain or changing the visuals displayed by the web page is accessible to the attacker, these changes affect how the victim perceives the website. Thus, the attacker may not consider them. Shirazi et al. do not directly mention the concept of constraints. However, in their mutation strategy, they specified that the attacker could only mutate to values that appeared to other phishing instances.

Table 1. Threat Model Comparisons

Attacker's	Our threat model	Shirazi et al. threat model
Goal	Decrease recall of the model	Decrease recall of the model
Knowledge	Selected features	Selected features Model type
Influence	Manipulate features to avoid detection	Manipulate features to avoid detection
	Limited attempts	Unlimited attempts
Control	All features	All features except domain
Constraints	The attacker can only change the phishing instance slightly	The attacker can only mutate to values that appeared in other phishing instances

3.2 Adversarial Sampling for Phishing

In this section, we describe how we attempt to simulate an attacker's approach by mutating existing phishing URLs. This mutation strategy is applicable to textual features. The attacker is allowed a fixed number of attempts for each instance, where we call the number of allowed attempts generation number. If any of the attempts manage to fool the classifier, the instance is considered a false negative. Fig.2 illustrates the generation number. We report the final recall, which shows how the model was affected.

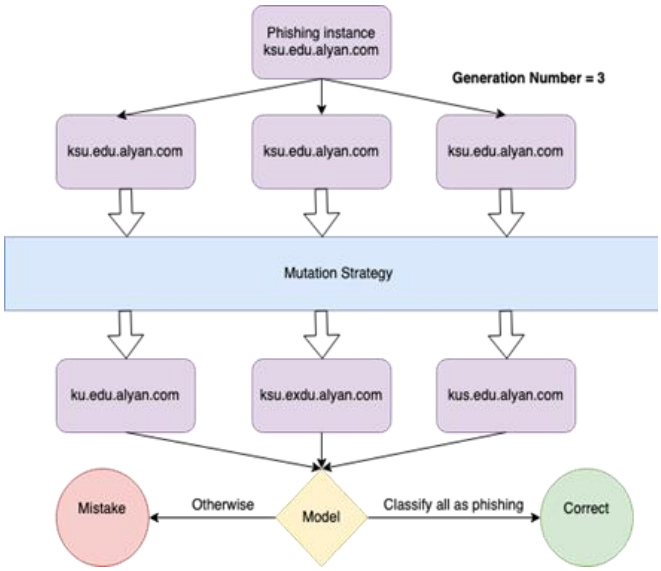

Fig. 2: Illustration of Generation Number

Mutations to the URLs have a cost that is hard to measure, which is how the target of the phishing attack would perceive the phishing URL after the mutation. Even a small change could make the target more suspicious. We need to measure such possible mutations. We propose the following conservative mutation strategy. We allow the attacker to perform only one of the following mutation types: inserting a letter, removing a letter, or swapping any two adjacent letters. The mutation is allowed only if the URL is parsed as a valid URL. To model the attacker's behavior, we use a uniform distribution where we assume that the attacker uniformly chooses from the possible mutation types. The attacker also uniformly chooses the letter to insert, the letter to delete or the adjacent letters to swap. We assume that the attacker can repeatedly apply this mutation strategy, where we call the number of repetitions the mutations number. Fig.3 illustrates our mutations strategy. It is also possible, although unlikely that such mutations will result in an already registered domain. However, we ignore this in our analysis.

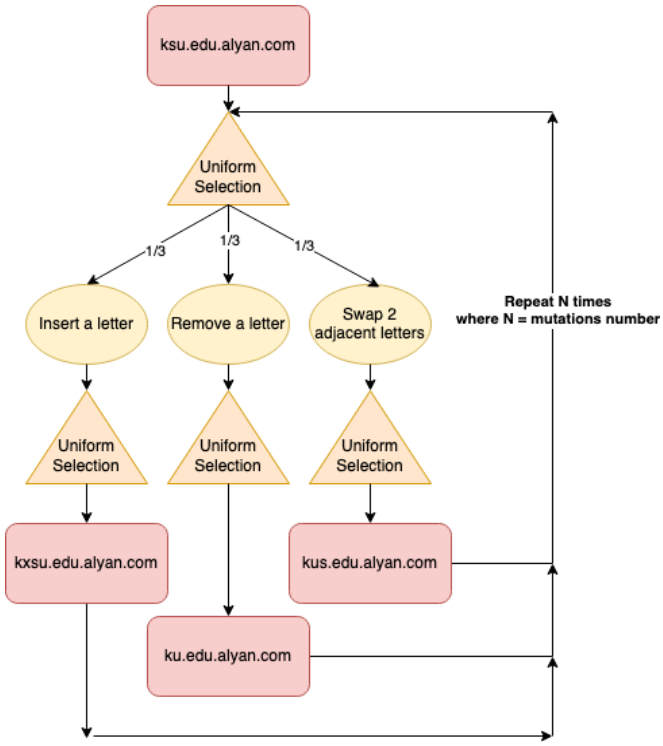

Fig. 3: Illustration of our Mutations Strategy and Mutations Number

4 Experiments and Results

4.1 Statistical Measures

Four popular statistical measures for classification models are precision, recall, accuracy, and F-measure [18]. They are calculated as follows:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (3)$$

$$F - measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4)$$

where TP, TN, FP, and FN are the occurrence number of the model prediction of true positives, true negatives, false positives, and false negatives, respectively.

Table 2, which is called the confusion matrix, shows how TP, TN, FP, and FN relate to the prediction and the actual value. In this paper, we consider phishing as positive and legitimate as negative.

Table 2. Confusion Matrix

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

We note that high precision is preferable in situations where false positives are not preferred, while a high recall is preferable when false negatives are not preferred. In the case of website phishing detection, high precision means a lower number of legitimate websites classified as phishing websites. In comparison, high recall means a lower number of phishing websites that were classified as legitimate. Both precision and recall are essential, depending on the usage scenario. For example, it may be preferable to have high precision on personal devices, while on the other hand, for some firewalls, it may be preferable to have a high recall. F-measure is the harmonic mean of precision and recall. Finally, we note that in our adversarial scenario, the goal of the attacker, as we will discuss later, is to decrease recall. This makes recall the main performance metric of this paper. However, we also use an accuracy metric for benchmarking the domain-exclusive model.

4.2 Experiments Setup

In this paper, we perform four experiments. In these experiments, we continue our previous work and utilize PUCNN. We also use the same training, validation, and testing datasets that were randomly split from the preprocessed MUPD dataset [2]. In the first experiment, we preprocessed each URL and converted it to its host string (Usually a domain or an IP). We report the accuracy of the testing dataset, and we compare it to the original URL-based PUCNN results. The goal of this experiment is to show how Shirazi et al. [6] threat model is not effective in this case as the model is based only on domains (or more accurately host strings). In Shirazi et al. threat model the attacker cannot modify the domain. Shirazi et al restriction may be useful in some cases as it can simplify the analysis.

Next, we discuss experiments 2-4. In these experiments, we evaluated the original URL-based PUCNN from [2] against the phishing instances in the testing dataset after mutations. We note that we only consider classification correct if all instances generated from the same instance are classified correctly. We performed the proposed mutations strategy with all combinations of specific values of mutations number and generation number. Table 3 lists the values we experimented with. However, because of the huge increase in the generation number, the number of instances increases. To do the test, we performed sampling based on the generation number instead. The sample size we used is the size of the test dataset divided by the generation number.

Table 3. Experiments 2-4 Variables

Mutations Number	Generation Number
1, 5, 10, 100	1, 10, 100, 1000

The main difference among experiments 2-4 is the location of the mutations. In experiment 2, we only performed the mutations on the domain part of the URL. Whereas, in experiment 3, we performed the mutation in the non-domain part of the URL. Finally, in experiment 4, we performed the mutations on the whole URL. We implemented the whole URL mutations as a random uniform choice between non-domain and domain mutations. This means even if the non-domain part is long, and the domain part is short they are getting mutated at the same rate. Table 4 summarizes the experiments. Using these experiments, we can find how the evaluation is affected based on the mutation location. In addition, we note that non-domain mutations are cheaper for the attacker because the attacker can register only one domain. Whereas in the other cases, the attacker needs to register each uniquely generated domain.

4.3 Results

In this section, we first show benchmarks between the accuracies of URL-based PUCNN and domain-based PUCNN. Then, we show how URL-based PUCNN performs under our adversarial sampling.

Table 4. Setup of Experiments 2-4

Ex #	Mutation Location
2	Domain part
3	Non-domain part
4	Whole URL

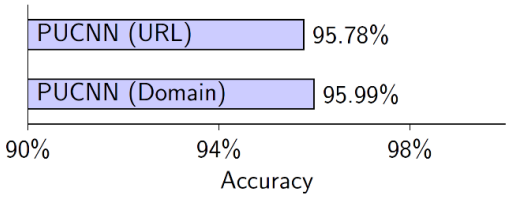

Fig. 4: Accuracy of PUCNN

From Fig.4 we note that PUCNN achieved better accuracy when using domains directly instead of URLs.

We can see how PUCNN performed well under adversarial sampling in tables 5, 6, and 7. PUCNN had the best results when the mutations were exclusive to the domain. On the other hand, PUCNN had the worst results in the experiments when the mutations were allowed on the whole URL.

In general, it can be seen from the results that PUCNN performed well when the mutations number and generation number are small. Small generation numbers are what we expect in practice. For example, a high generation number risks the attack discovery. While having high mutations number make the URL very different from what it was, which may make the user doubts the URL. These results imply that PUCNN is robust in this scenario. However, this does not cover scenarios where the attacker comes up with a new URL or uses a more complex mutation strategy.

Table 5. Domain Mutations Recalls

Mut # / Gen #	1	10	100	1000
1	94.6352	89.1628	84.3554	82.6464
5	93.4983	82.5548	73.0661	64.2082
10	92.7078	79.4133	69.3391	58.7852
100	89.7365	74.1940	63.1636	51.1931

Table 6. Non-domain Mutations Recalls

Mut # / Gen #	1	10	100	1000
1	94.3376	87.8781	82.3619	80.6941
5	91.9144	76.6.076	61.8202	52.0607
10	89.9788	70.0559	53.7161	43.8178
100	86.9918	60.4190	43.2069	35.1410

Table 7. Whole URL Mutations Recalls

Mut # / Gen #	1	10	100	1000
1	94.3977	87.1544	79.5450	75.7050
5	92.0442	73.9211	55.4496	38.8286
10	89.3242	63.3829	38.8516	24.0781
100	80.1047	29.5000	8.8841	3.9046

5 Conclusion

In this work, we presented a method for evaluating phishing detection models in adversarial situations by adversarial sampling attacks. We found some limitations such as the exclusion of domain modifications and non-applicability for models that utilize the URL directly. Additionally, all the studied models did not perform well in the evaluation. This may be because the attacker was unrestricted in the proposed threat model, as the attacker had unlimited access to the prediction function. To address these limitations, we proposed a more restricted adversarial scenario where the attacker has limited access to the prediction function.

To evaluate the adversarial scenario, we proposed a text-based mutation strategy which we used to perform adversarial sampling attacks. This mutation strategy is applicable to models that utilize the URL directly. This mutation strategy is parameterized where the parameters tune the attacker restrictions. Finally, we evaluated PUCNN, our previous contribution, which is state-of-art model that utilizes the URL directly as it is based on a character level CNN. We have found that PUCNN performed well for low parameters which indicate a more restricted attacker.

References:

[1] E. E. H. Lastdrager, “Achieving a consensual definition of phishing based on a systematic review of the literature,” *Crime Sci.*, vol. 3, no. 1, p. 9, Dec. 2014, doi: 10.1186/s40163-014-0009-y.

[2] A. Al-Alyan and S. Al-Ahmadi, “Robust URL Phishing Detection Based on Deep Learning,” *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 7, pp. 2752–2768, Jul. 2020, doi: 10.3837/tiis.2020.07.001.

[3] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, “Object recognition with gradient-based learning,” *Springer*, 1999.

[4] B. Biggio and F. Roli, “Wild Patterns,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Oct. 2018, pp. 2154–2156, doi: 10.1145/3243734.3264418.

[5] M. A. Adebowale, K. T. Lwin, and M. A. Hossain, “Intelligent phishing detection scheme using deep learning algorithms,” *J. Enterp. Inf. Manag.*, vol. ahead-of-p, no. ahead-of-print, Jun. 2020, doi: 10.1108/JEIM-01-2020-0036.

[6] 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)*, vol. 11559 LNCS, no. July, 2019, pp. 83–101.

[7] "Keyword research, competitor analysis, & website ranking." https://alexa.com (accessed Oct. 07, 2018).

[8] "PhishTank - join the fight against phishing." https://www.phishtank.com/ (accessed Oct. 24, 2018).

[9] R. M. Mohammad, F. Thabtah, and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," *2012 Int. Conf. Internet Technol. Secur. Trans. ICITST 2012*, pp. 492–497, 2012.

[10] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based Associative Classification data mining," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 5948–5959, Oct. 2014, doi: 10.1016/j.eswa.2014.03.019.

[11] C. L. Tan, "Phishing Dataset for Machine Learning: Feature Evaluation." 2018, doi: 10.17632/h3cgnj8hft.1.

[12] "Download list of top 10 million domains based on open data from common crawl & common search." https://www.domcop.com/top-10-million-domains (accessed Jun. 28, 2019).

[13] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Syst. Appl.*, vol. 117, pp. 345–357, Mar. 2019, doi: 10.1016/j.eswa.2018.09.029.

[14] W. Wang, F. Zhang, X. Luo, and S. Zhang, "PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks," *Secur. Commun. Networks*, vol. 2019, pp. 1–15, Oct. 2019, doi: 10.1155/2019/2595794.

[15] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, pp. 1–28, Sep. 2011, doi: 10.1145/2019599.2019606.

[16] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, pp. 649–657, Sep. 2015, [Online]. Available: http://arxiv.org/abs/1509.01626.

[17] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-Aware Neural Language Models," *30th AAAI Conf. Artif. Intell. AAAI 2016*, pp. 2741–2749, Aug. 2015, [Online]. Available: http://arxiv.org/abs/1508.06615.

[18] A. Tharwat, "Classification assessment methods," *Appl. Comput. Informatics*, vol. 17, no. 1, pp. 168–192, Jan. 2021, doi: 10.1016/j.aci.2018.08.003.