Backdoor Trigger Detection Using Adversarial Perturbation for Deep-learning Phishing Detector

Koko Nishiura*, Tomotaka Kimura*, and Jun Cheng*
*Graduate School of Science and Engineering, Doshisha University, Kyoto, Japan
Email: tomkimur@mail.doshisha.ac.jp; jcheng@ieee.org

***Abstract*—In this paper, we propose a backdoor trigger-detection method using an adversarial perturbation for deep-learning phishing detectors. Recently, backdoor attacks have become a serious threat to deep-learning–based detectors. To counter backdoor attacks, the proposed method uses the adversarial perturbation generated by the universal adversarial perturbation method. Using this approach, the proposed method is not only able to detects backdoor attacks, but can also detect the location where the trigger has been inserted. Therefore, once we have several URLs containing the trigger, we can identify the triggers themselves, allowing the attacker's characteristics to be analyzed. Through experiments using a dataset of phishing site URLs, we show that the proposed method can counter backdoor attacks.**

***Index Terms*—Adversarial perturbation, Backdoor attack, Phishing detection**

I. Introduction

Recently, phishing scams have become a major threat [1] and cause of financial damage. Specifically, phishing scams are attempts to steal personal information through phishing websites that mimic legitimate websites. As a countermeasure against phishing scams, deep-learning–based phishing detection methods have attracted attention because of their high performance [2]. However, to construct a highly accurate detector, a large amount of data must be used for training, and the computational resources required for such processing are significant.

To solve the problem of the computational resources, learning outsourcing services have emerged that can train the deep-learning detector on their cloud servers. Vendors providing such services are called learning providers. Learning providers can create deep-learning models at a low cost using their cloud servers, which solves the problem of inadequate computational resources. However, vulnerable deep-learning models may be intentionally provided because the learning is performed by a third-party vendor.

An attack that learning providers can launch is a backdoor attack [3] (Fig. 1). In a backdoor attack, when training a machine-learning model, the learning provider with malicious intent adds poisoned data containing triggers to the data sent by the customer. The trained machine-learning model is then sent to the customer. The customers do not know that training took place using poisoned data, and hence they do not suspect that a backdoor attack has been set up and they trust this model. However, traps have been set in this model so that poisoned data trigger them and cause misclassification.

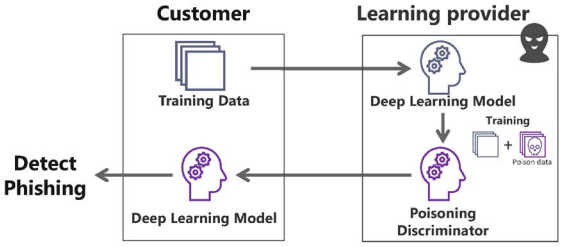


Fig. 1: Backdoor Attacks during Outsourcing.

To date, a countermeasure against backdoor attacks has been proposed [4]. In [4], a backdoor attack is detected by using the characteristic that the class of a backdoor attack instance does not change even if a slight perturbation is injected. Specifically, an adversarial perturbation is calculated in advance and the change in the class is examined when the adversarial perturbation is injected. In [4], the effectiveness of this adversarial perturbation was demonstrated for image processing.

In this paper, we propose an adversarial perturbation-based trigger-detection method for a phishing website detector. The proposed method not only detects backdoor attacks, but also the location where the trigger has been inserted. Therefore, once we have several URLs (Uniform Resource Locator) containing the trigger, we can identify the triggers themselves and hence analyze the attacker's characteristics. Through experiments using a dataset of phishing site URLs, we demonstrate that the proposed method can counter backdoor attacks.

The rest of the paper is organized as follows: In section II, we describe our proposed trigger detection method. Next, we evaluate the performance of our proposed method in section III. Finally, in section IV, we conclude the paper.

II. Proposed Trigger Detection Method

Figure 2 shows an overview of the proposed trigger-detection method. In the proposed method, when the trigger is embedded into a URL, the trigger is extracted from the URL using an adversarial example technique, i.e., the UAP (Universal Adversarial Perturbations) method [5]. UAP finds a perturbation vector $\boldsymbol{v}$ that causes the misclassification of a discriminator f for a large number of data points. More specifically, UAP finds a perturbation vector $\boldsymbol{v}$ such that $f(\boldsymbol{x}+\boldsymbol{v})$ and $f(\boldsymbol{v})$ are different for almost all data points.

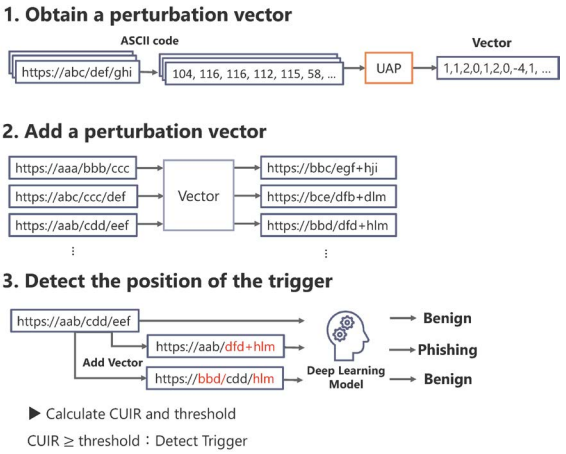


Fig. 2: Overview of Trigger Detection.

In the proposed method, for dataset $\mathcal{X}$, the perturbation vector $\boldsymbol{v}$ is obtained using UAP, and then the successive k elements of perturbation vector $\boldsymbol{v}$ are changed to 0 (Fig. 3). Here, let $\boldsymbol{v}_n$ denote the perturbation vector whose nth to $(n+k-1)$th elements are 0. For $\boldsymbol{v}_n$ $(n = 1, 2, \ldots)$, $C(\boldsymbol{v}_n)$ is calculated as follows:

$$C(\boldsymbol{v}_n) = \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}(f(\boldsymbol{x} + \boldsymbol{v}_n) = f(\boldsymbol{x}))}{|\mathcal{X}|}, \tag{1}$$

where $\mathbb{1}$ is the indicator function and $|\mathcal{X}|$ is the number of the elements in dataset $\mathcal{X}$. Moreover, $C(\boldsymbol{v}_n)$ represents the ratio of misclassification that does not occur due to the perturbation vector $\boldsymbol{v}_n$. The proposed method takes this approach because changing the value of the part containing the trigger is likely to cause misidentification to a different class. In other words, if the part containing the trigger is not perturbed, misidentification to a different class is less likely to occur and the value of $C(\boldsymbol{v}_n)$ should be high.

To detect the position of the trigger, we calculate the average μ and standard deviation σ of $C(\boldsymbol{v}_n)$ $(n = 1, 2, \ldots)$. If $C(\boldsymbol{v}_n)$ is greater than $\mu + \alpha\sigma$, the proposed method guesses that the trigger is located at position n, where α is a threshold parameter.

As for the rationale behind the detection of triggers in our proposed method, there might be a large distortion in the model's decision boundary for the elements at the input position of the trigger. UAP requires a perturbation vector $\boldsymbol{v}$ that causes misclassification for every element in $\mathcal{X}$, and the easiest way to cause a misclassification is to perturb it around the injected trigger location because the decision boundary has been forcibly distorted by the trigger injection. Based on this, the proposed method finds the trigger-injected location by forcing some of the elements of the perturbation vector to zero, that is, by removing the perturbation. When elements match the trigger, the elimination of the perturbation reduces the misclassification. Therefore, the value of $C(\boldsymbol{v}_n)$ increases when the perturbation of the triggering part is removed.

Fig. 3: Addition of $\boldsymbol{v}_n$.

III. Performance Evaluation

A. Setting

To demonstrate the effectiveness of the proposed trigger-detection method, we conducted experiments using the dataset provided in [6], which contains URL data from both phishing and benign sites. We used 38,272 URLs (16,112 phishing and 22,160 benign URLs) and divided them into two datasets: a training dataset and verification dataset. The training data consist of 32,272 URLs (14,912 phishing and 17,360 benign URLs) and were used to train discriminator f by the learning provider. By contrast, the verification data consist of 6,000 URLs (1,200 phishing and 4,800 benign URLs) and were used to verify whether discriminator f is vulnerable to a poison attack.

In the scenarios we consider in this evaluation, we assume that the training data are provided to a learning provider. The learning provider trains discriminator f using its computational resources. We also assume that the learning provider has malicious intent and launches a back door attack. Specifically, the learning provider chooses some URLs from the training data and embeds the trigger into these URLs. Therefore, the learning provider trains discriminator f using the training data and trigger-embedded data. In our experiments, there were two scenarios depending on the trigger positions. One was inserted from positions 140 to 145, and the other was inserted from positions 160 to 165. The trigger was "AAAAAA".

As discriminator f, we used a four-layer neural network with an input layer of 200 nodes; the intermediate layer had two dense layers with 64 and 32 nodes, and the output layer had two nodes. To input a URL into the discriminator, the URL was converted into a numeric vector using ASCII code. Because this vector is of fixed length, if the length of the URL is short, zero padding was applied. By contrast, if the length of the URL was larger than 200, the end of the URL was ignored. After the learning provider trained the discriminator f, it delivered the discriminator f to the customer. In the scenarios, when the customer receives the discriminator f, it checks the accuracy of the discriminator using the training data. For the training data used in our experiments, the accuracy of the discriminator was 90.77%. At first glance, discriminator f performs well because the training data does not contain trigger data. Therefore, the customer is unaware that a backdoor attack has been launched, and uses discriminator f to detect phishing URLs.

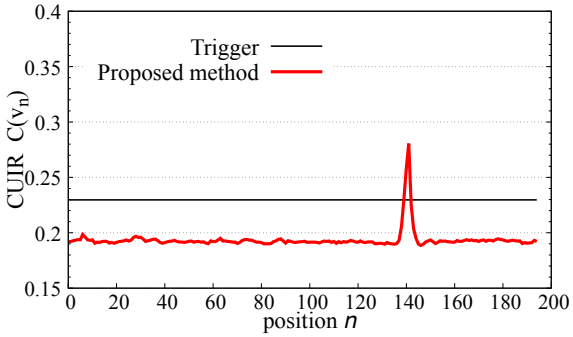
After the initial evaluation of the discriminator, the customer collects URLs to be evaluated by the detector to determine whether a backdoor attack has been launched. In the experiment, half of the phishing URLs (i.e., 600 URLs) in the verification data were injected with the trigger, and they were converted into poisoned URLs. We divided the verification data (600 phishing, 600 poisoned, and 4800 benign URLs) into 20 datasets $\mathcal{X}_i$ $(i = 1, 2, \ldots, 20)$. The customer checks whether there is a backdoor in discriminator f using dataset $\mathcal{X}_i$. Specifically, the perturbation vector $\boldsymbol{v}$ is calculated for $\mathcal{X}_i$, and then CUIR (Clean URL Identification Rate) $C_i(\boldsymbol{v}_n)$ for $\mathcal{X}_i$ is calculated using (1). The parameter k is set to 6.

As a performance index, we used the mean CUIR $\overline{C}(\boldsymbol{v}_n)$, which is the average of the 20 datasets. The mean CUIR $\overline{C}(\boldsymbol{v}_n)$ is calculated as $\overline{C}(\boldsymbol{v}_n) = (1/20)\sum_{i=1}^{20} C_i(\boldsymbol{v}_n)$. In this evaluation, the threshold α was set to 3.

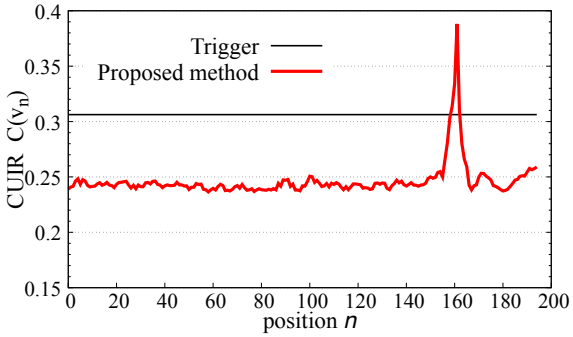
B. Results

First, we present the performance of the proposed method in detecting triggers. Figure 4 shows the mean CUIR $\overline{C}(\boldsymbol{v}_n)$ as a function of position n. From Figs. 4 (a) and (b), we can observe that the mean CUIR is larger in the areas where the trigger has been inserted, i.e., positions 140–145 and 160–165, respectively. Moreover, the threshold is exceeded only in the areas where the trigger has been inserted. These results indicate that the proposed method can detect the presence of backdoor attacks. Furthermore, the presence of triggers in the areas above the threshold can also be detected.

Next, we verify whether the proposed method causes false positives when a backdoor attack has not been launched. Specifically, we consider the situation in which the learning provider has no bad intentions and no poisoned URLs are used to train discriminator g. Figure 5 shows the mean CUIR $\overline{C}(\boldsymbol{v}_n)$ for discriminator g. In this figure, the mean CUIR is almost the same for all n, and unlike the results for discriminator f, no n exceeds the threshold value. This result confirms that the proposed method does not falsely detect triggers in the absence of a backdoor attack.



(a) Trigger position: 140-145



(b) Trigger position: 160-165

Fig. 4: Mean CUIR $\overline{C}(\boldsymbol{v}_n)$ for Poisoning Discriminator f.

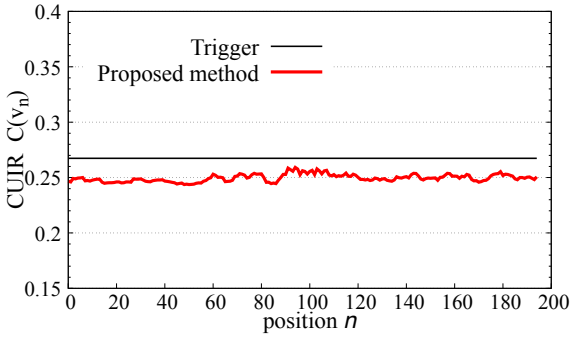


Fig. 5: Mean CUIR $\overline{C}(\boldsymbol{v}_n)$ for Clean Discriminator g.

IV. Conclusion

In this paper, we proposed a countermeasure against backdoor attacks for deep-learning–based phishing detectors. Using an adversarial perturbation, the proposed method detects the positions of the trigger. Through experiments using a dataset of phishing site URLs, we demonstrated that the proposed method can identify the positions of injected triggers.

Acknowledgment

This research was supported by JSPS KAKENHI (20H04184).

References

[1] H. S. Lallie, L. A. Shepherd, J. R. Nurse, A. Erola, G. Epiphaniou, C. Maple, and X. Bellekens, "Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic," *Computers & security*, vol. 105, p. 102248, 2021.

[2] A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, "A comprehensive survey of ai-enabled phishing attacks detection techniques," *Telecommunication Systems*, vol. 76, pp. 139–154, 2021.

[3] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[4] M. Xue, Y. Wu, Z. Wu, Y. Zhang, J. Wang, and W. Liu, "Detecting backdoor in deep neural networks via intentional adversarial perturbations," *Information Sciences*, vol. 634, pp. 564–577, 2023.

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1765–1773.

[6] J. Feng, L. Zou, O. Ye, and J. Han, "Web2vec: Phishing webpage detection method based on multidimensional features driven by deep learning," *IEEE Access*, vol. 8, pp. 221 214–221 224, 2020.