Received: 23 September 2020 | Revised: 18 April 2021 | Accepted: 11 May 2021

DOI: 10.1002/int.22510

RESEARCH ARTICLE

WILEY

Advanced evasion attacks and mitigations on practical ML-based phishing website classifiers

Fu Song[1] | **Yusi Lei**[1] | **Sen Chen**[2] | **Lingling Fan**[3] | **Yang Liu**[4]

[1]School of Information Science and Technology, ShanghaiTech University, Shanghai, Pudong, China

[2]College of Intelligence and Computing, School of Cybersecurity, Tianjin University, Tianjin, Jinnan, China

[3]College of Cyber Science, Nankai University, Tianjin, Jinnan, China

[4]School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

Correspondence
Fu Song, School of Information Science and Technology, ShanghaiTech University, No. 393 Huaxia Middle Road, Shanghai 201210, Pudong, China
Email: songfu@shanghaitech.edu.cn and songfu1983@shanghaitech.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Number: 62072309

Abstract

Machine learning (ML) based classifiers are vulnerable to evasion attacks, as shown by recent attacks. However, there is a lack of systematic study of evasion attacks on ML-based anti-phishing detection. In this study, we show that evasion attacks are not only effective on practical ML-based classifiers, but can also be efficiently launched without destructing the functionalities and appearance. For this purpose, we propose three mutation-based attacks, differing in the knowledge of the target classifier, addressing a key technical challenge: automatically crafting an adversarial sample from a known phishing website in a way that can mislead classifiers. To launch attacks in the white- and gray-box scenarios, we also propose a sample-based collision attack to gain the knowledge of the target classifier. We demonstrate the efficacy of our evasion attacks on the state-of-the-art, Google's phishing page filter, achieved 100% attack success rate in less than one second per website. Moreover, the transferability attack on BitDefender's industrial phishing page classifier, TrafficLight, achieved up to 81.25% attack success rate. We further propose a similarity-based method to mitigate such evasion attacks, Pelican, which compares the similarity of an unknown website with recently detected phishing websites. We demonstrate that Pelican can effectively

detect evasion attacks, hence could be integrated into ML-based classifiers. We also highlight two strategies of classification rule selection to enhance the robustness of classifiers. Our findings contribute to design more robust phishing website classifiers in practice.

KEYWORDS
adversarial attacks, adversarial sample detection, machine learning, mutation, phishing website

1 | INTRODUCTION

Web phishing is a major cyber threat and have reached a record volume.[1] It has been widely used to steal personal information such as login credentials. In addition, people nowadays prefer online shopping, which aggravates the harms of phishing attacks in practice. Consequently, phishing attackers are able to gain a (vast) amount of money via phishing websites.[2] To detect phishing attacks, various anti-phishing solutions have been proposed, such as blacklists,[3] similarity-,[4] and machine learning (ML)-based[5] approaches. However, blacklist is ineffective in detecting 0-day attacks[3]; similarity-based approaches being capable of detecting 0-day attacks, have limited scalability and accuracy.[6,7] ML-based ones are not only scalable and accurate, but can also detect 0-day attacks, hence have been widely studied[5,8–10] and deployed in industrial browsers (e.g., *Chrome* and *Edge*[11]).

Prior research shows that ML-based classifiers are vulnerable to evasion attacks, for example.[12–19] Such attacks have been extensively studied in image recognition and malware detection, but little has done in anti-phishing. This is potentially due to the new challenge in this domain: unlike *adversarial images* which only need to preserve the appearance and *adversarial malware* which only need to preserve the functionality, adversarial phishing websites have to preserve appearance and functionalities *simultaneously*, to be more effective for web phishing.[5] Hence, it becomes imperative to understand whether it is possible to effectively launch evasion attacks on ML-based anti-phishing classifiers with limited knowledge of the target classifier, while preserving the functionalities and appearance of the webpages.

In this study, we show that evasion attacks are not only effective on practical ML-based classifiers, but can also be launched without destructing functionalities and appearance of phishing webpages, in all white-, gray-, and black-box scenarios. For this purpose, we propose advanced evasion attacks, differing in the knowledge (e.g., classification rules, features, and weights) of a target classifier obtained by the adversary, namely, white-box attack with full knowledge, gray-box attack with partial knowledge, and black-box attack without any knowledge, of the ML-based classifier under attack. Specifically, to effectively and efficiently craft adversarial samples from known phishing websites, we propose three mutation mechanisms, each of which iteratively mutates a sample from a known phishing website according to the knowledge of the adversary until the target classifier is misled. To launch attacks in white- and gray-box scenarios, it is necessary to gain partial or full knowledge of the classifier under attack. Therefore, we propose a sample-based collision attack for inferring classification rules by leveraging the data collected from legitimate and phishing websites. In the white-box

attack setting, where (almost) full knowledge of the target classifier is obtained, we use a greedy strategy to choose which features to delete from or add into a sample to maximally decrease the decision score at each mutation step. In the gray-box attack setting, where only partial knowledge of the target classifier is obtained, we first iteratively delete all the features of known classification rules that can reduce the decision score, and if necessary, we add classification rules that do not exist in the original sample but can reduce the decision score. In the black-box setting, where no classifier information is exposed except that the adversary can query the classifier and obtain the decision score, based on our insights of existing ML-based classifiers, we first delete DOM nodes (i.e., nodes in HTML Document Object Model) from the sample and then add DOM nodes into the sample, guided by the decision scores.

To evaluate our attacks, we examined both academic and industrial ML-based tools (e.g., CANTINA+,[8] Off-the-Hook,[20] δPhish,[9] the tool developed by Ubing et al.,[10] Monarch[21] and Google's phishing page filter (GPPF)[5]) and industrial proprietary tools (Bitdefender TrafficLight, Netcraft Anti-Phishing Extension, and 360 Internet Protection). Except for GPPF, all the other tools either are not publicly accessible or cannot output the decision score for each input which is required even in our black-box scenario. Thus, we demonstrate our attacks on the state-of-the-art ML-based classifier GPPF. We emphasize that many ML-based classifiers such as CANTINA+, Off-the-Hook, and Monarch use the similar idea as GPPF, differ in features, architectures, and ML algorithms. Therefore, our attacks could be used for these ML-based classifiers. GPPF uses a logistic regression learning algorithm to train a classification model and is effective in phishing warnings.[22] It has been deployed in both Chrome and Firefox,[23] owning billions of users. In terms of false positive rate and accuracy, GPPF is also the best one among the 14 tools including similarity-, heuristic- and ML-based ones.[20] To show that our black-box attack is generic and effective for other practical ML-based classifiers, we launch transferability attacks on the industrial tool Bitdefender TrafficLight, which owns the most users and stars in the add-ons of Firefox. Furthermore, Bitdefender TrafficLight is proprietary without any publicly available information about the internal design and implementations, hence completely black-box. We also tried to launch transferability attacks on other anti-phishing tools, but failed to get their classification results *automatically*. In summary, our attacks have 4 prominent advantages:

- Effectiveness: Our attacks achieve 100% attack success rate on GPPF in all the white-, gray-, and black-box scenarios.
- Efficiency: Adversarial samples are crafted in less than one second per seed on average.
- Transferability: The transferability attack success rate on Bitdefender TrafficLight is up to 81.25%.
- Nondestructiveness: All the crafted adversarial samples have the same functionalities and appearance as their original ones, where appearance is measured by two common distortion criteria of images: mean-squared error (MSE)[24] and mean-absolute error (MAE).[25]

To mitigate the real threats posed by evasion attacks, we propose a similarity-based method, named Pelican, which compares the similarity of DOM trees between an unknown one and recently detected phishing websites. This method neither modifies the target classifier nor relies on specific properties of the classifier, so it could be used to protect a wide range of classifiers as a pre-filter. We remark that Pelican is used to efficiently detect evasion attacks rather than general phishing attacks as done by existing work.[4,26] Furthermore, to mitigate collision attacks and enhance the robustness of classifiers, we

identify weaker classification rules and propose two strategies: *removing single classification rules and subset classification rules from classifiers*. We also suggest adding more *appearance-related features* into these classification rules to increase the robustness of the classifier. Our findings contribute to design more robust phishing website classifiers in practice. We evaluate Pelican on 915 adversarial samples (one sample per seed and scenario). Pelican is able to detect all these samples. We also evaluate classification rule selection strategies using our black-box attacks and demonstrate their weakness, but have no side-effect on checking original phishing/benign websites when subset rules or single rules are disabled.

In summary, we make the following contributions.

- Novel evasion attacks on ML-based web phishing classifiers, which are effective, efficient, and nondestructive.
- A sample-based collision attack that can effectively infer more features/rules for gray- and white-box evasion attacks.
- Evaluation of our attacks which achieved 100% attack success rate on the state-of-the-art classifier `GPPF` and achieved up to 81.25% transferability attack success rate on Bitdefender TrafficLight, in the real world.
- A defense method, Pelican, to mitigate evasion attacks, and two strategies (i.e., removing single classification rules and subset classification rules) to enhance the robustness of the classifiers.
- A data set[27] including 3000 phishing websites, 15,000 legitimate website URLs, and 2,566,834 phishing URLs to foster further research such as phishing website detection/defense and measurement.

To the best of our knowledge, this is the first systematic study of evasion attacks on practical ML-based phishing website classifiers considering all the white-, gray- and black-box scenarios, the first defense work against evasion attacks in this domain and the first evaluation of the robustness of single classification rules and subset classification rules under the adversarial environment.

2 | BACKGROUND

2.1 | Website and webpage structure

A website consists of webpages for presenting information to users. Webpages are accessed via their corresponding URLs. As shown in Figure 1, a webpage has a basic structure and multiple corresponding elements, representing in a tree hierarchy, called *DOM tree*. Specifically, a webpage has several kinds of nodes, called DOM nodes, including element (e.g., HTML element), text (e.g., text inside the HTML element), attribute (e.g., the attributes of each HTML element), and comment codes (no effect on webpage structure).

The children of an element node are text and attribute nodes. In addition, scripts such as `JavaScript` code are also an important part of webpages which can manipulate (e.g., *modify*, *delete*, and *add*) the DOM tree. In this study, we consider JavaScript only, other scripting languages can be handled similarly. The appearance and functionalities of a webpage are determined by the combination of elements, texts, attributes, and scripts.

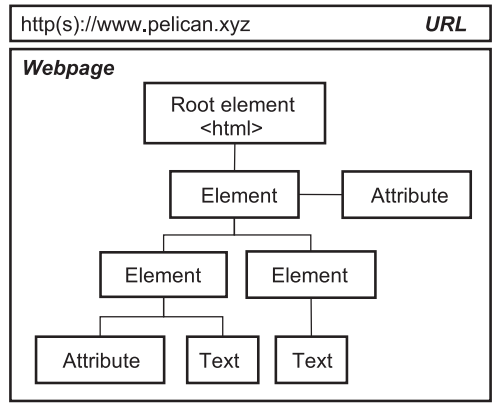

FIGURE 1 Structure of a webpage

2.2 | Web phishing

Web phishing, a type of social engineering attack, is often used to steal user information, such as login credentials and payment data.[28] It occurs when an attacker masquerades the phishing website as a trusted entity, both of the webpages and URLs are very close to the original legitimate ones. Typically, attackers first create a fake website and then distribute its URL to lure users to click the URL link, which redirects to a phishing website from untrusted servers to extract users' information. Two traditional strategies are often adopted in web phishing to distribute URL: *spoofed emails* and *fake websites*.[29] The former is an active attack, which sends numerous spoofed emails to various users and the spoofed emails lure users to click links in the email that redirects to a phishing website. In contrast, the latter is a passive attack, which distributes the URL via other ways (e.g., bulletin board systems, messaging services) and waits for users' visit of the phishing website. Attackers aim to make the phishing URLs and visual appearances of webpages as similar as possible to the targeted victim's.

2.3 | ML-based anti-phishing

Generally, ML-based anti-phishing methods extract a set of features of text, image, or URL information from DOM trees of authentic and/or phishing websites. A set of classification rules obtained from the learning algorithms are used for phishing detection. According to the existing ML-based anti-phishing solutions, we define a generic *ML-based phishing classifier* c as a tuple

$$c \triangleq (R, \pi, \tau),$$

where R is a finite set of classification rules, π is a function used to compute the final decision score of a given website, and τ is the threshold. Classification rules are described as follows:

$$\begin{aligned} \text{Rule: } r &= (id_r, F_r, w_r) \\ \text{Feature: } f &= (n_f, v_f) \\ \text{Node: } n &= (\mathtt{n}_{\mathtt{type}}, \mathtt{n}_{\mathtt{tag}}, \mathtt{n}_{\mathtt{children}}, \mathtt{n}_{\mathtt{v}}) \end{aligned}$$

A classification rule r consists of an identity id_r, a set of features F_r, and a weight w_r contributing to the final decision score. We denote by $W = (w_r)_{r \in R}$ the set of weights. A feature f has a value v_f which is computed from a set n_f of DOM nodes of webpages. The values of features in F_r determine the value of r. For example, a feature `PageHasForms` examines whether a webpage has a form node, and its value is either 1 or 0. A DOM node has a type $\mathtt{n}_{\mathtt{type}}$ (e.g., element, attribute, and text) and a tag name $\mathtt{n}_{\mathtt{tag}}$ (e.g., "div"). An element node has children $\mathtt{n}_{\mathtt{children}}$ that are attribute and text nodes. An attribute or text node has a value $\mathtt{n}_{\mathtt{value}}$. For example, a "type" attribute node has the value "password" and a text node has the value "Hello World!". In practice, classification rules are usually designed according to domain knowledge of experts, while their weights are trained from some data set using ML algorithms.

A classification rule r is hit by a website p, denoted by $p \vDash r$, only if all the values of its features in p are nonzero. To check a website p, it computes the decision score $\mathtt{score}_c(p)$ based on weights of classification rules hit by p using the function π. It is considered to be a phishing one if the decision score exceeds a predefined threshold, that is, $\mathtt{score}_c(p) \geq \tau$, otherwise, it is regarded as a legitimate one. Although, classifiers may differ in concrete anti-phishing solutions, our definition is generic and is able to capture the cores of many published ML-based solutions[5,7–10,20,21]

3 | THREAT MODEL AND OVERVIEW

In this section, we propose our threat model, and then introduce the overview of our study.

3.1 | Threat model

We assume that an attacker wants to craft a sample from a desired phishing website that has the same functionalities and appearance, but is misclassified as benign by the target classifier; and (s)he can query to the classifier as many times as possible. This is feasible in practice, as limiting the number of query times will restrict the usage of the classifier.

We consider three attack scenarios: white-, gray-, and black-box. White-box means that the adversary has almost *full* knowledge of the target classifier (i.e., both classification rules and their weights). White-box scenario is not always tractable in practice, but it is feasible in practice. For instance, the classifier Bitdefender TrafficLight is proprietary without any publicly available information about the internal design and implementations, hence it is infeasible to launch white-box attack against TrafficLight. However, client-side classifiers such as `GPPF` do run in the environment that may be completely controlled by the adversary. Though labor-intensive, the adversary always can find a way to observe its running and gather exploitable information by, e.g., reverse engineering[30] and collision attacks.[31] Consequently, the adversary will have full or partial knowledge of the classifier under attack, allowing (s)he to launch a white-box evasion attack on the target classifier. Therefore, white-box scenario is feasible in practice.

Gray-box means that the adversary has access to classification rules, but not their weights, of the classifier under attack, since the weights of classifiers may be frequently updated in practice. For instance, Google retrains `GPPF` daily on the server-side using samples from classification data collected over the last three months and updates the weights of `GPPF` on the client-side.[5] By launching gray-box attacks, the adversary does not need to infer the weights of

the classification rules, hence boosting attack efficiency. Remark that it may be impossible to obtain full classification rules in practice. For instance, we cannot obtain full classification rules of GPPF. We argue that essential and practically easy obtained classification rules are sufficient to launch a successful attack. Indeed, our attack against GPPF under the gray-box scenario only has access to partial classification rules rather than full classification rules.

In the more realistic black-box setting, the adversary does *not* have any knowledge of the target classifiers, for example, the classifier Bitdefender TrafficLight. Even under such a realistic scenario, we will show that attacks are still possible via transferability attacks.

In all three scenarios, the adversary does not know any information about the training data set and training algorithm. We also assume that the classifier outputs the decision score of each input sample, which is a widely used assumption even in black-box scenario.[12,32,33] We will demonstrate that our approach is effective in transferability attacks, allowing us to attack classifiers that only output the category of each input instead of the decision score.

3.2 | Approach overview

Figure 2 shows the overview of our work on attacks and defenses of phishing classifiers: including advanced evasion attacks in white-, gray-, and black-box scenarios and an effective similarity-based method for evasion attack detection. We first propose a sample-based collision to get the knowledge of classifiers such as classification rules and their corresponding weights (marked ①). Second, given the desired phishing website that can be successfully classified as phishing by the classifier under attack, according to the knowledge the attacker obtained, (s)he leverages evasion attacks to craft adversarial samples that mislead the classifier (marked ②).

In the defense part, when the original one is identified as phishing by the classifier, its webpage structures are extracted (marked ③) for evasion attack detection. We propose an effective similarity-based method named Pelican (marked ④), which compares the similarity of DOM trees between an unknown one with recently detected phishing websites. If the similarity exceeds a pre-defined threshold, we regard the input website as a phishing one constructed by evasion attacks, but without invoking the ML-based classifier.

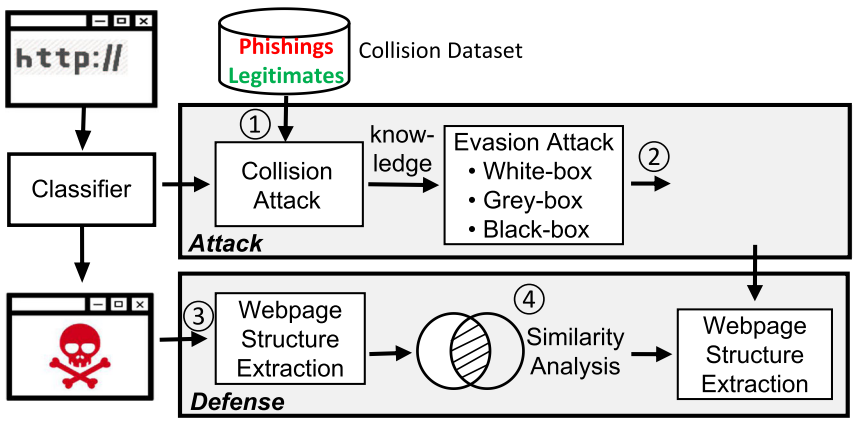

FIGURE 2 Overview of our study [Color figure can be viewed at wileyonlinelibrary.com]

We further come up with two classification rule selection strategies for enhancing the robustness of the classifiers and increasing the attack cost.

4 | ADVANCED EVASION ATTACKS

In this section, we propose a class of advanced evasion attacks under three different attack scenarios. Given a desired phishing website p, an *evasion attack* on a target classifier $c = (R, \pi, \tau)$ is to craft an adversarial sample mutated from p that can mislead the classifier c, according to the knowledge of c the attacker obtained.

To explain our approach, we will use GPPF as the target classifier, which is trained by a gradient descent logistic regression learning algorithm. Recall that all the other tools we examined either are not publicly accessible or cannot output decision score (cf. Section 1), therefore, we demonstrate our generic methods on the state-of-the-art classifier GPPF. In GPPF $c_{\texttt{GPPF}} = (R, \pi, \tau), R$ $c_{\texttt{GPPF}} = (R, \pi, \tau), R$ consists of 2130 classification rules that rely upon 1,009 features in total and the decision score function π is defined as

$$\pi(x) = \frac{e^x}{1 + e^x} \quad \text{and} \quad x = w_{r_1} + \sum_{i=2}^{2130} \left(w_{r_i} \cdot \prod_{f \in F_{r_i}} v_f \right),$$

where

- the threshold τ is 0.5,
- for every $1 \leq i \leq 2130$, w_{r_i} is the weight of the classification rule r_i and F_{r_i} denotes the set of features that r_i relies upon,
- for every $1 \leq i \leq 2130$ and every feature $f \in F_{r_i}$, v_f denotes the value of feature f in a given webpage p.

4.1 | Sample-based collision attack

To launch a gray- or white-box attack on a target classifier, it is necessary, but nontrivial, to obtain full or partial information of the classifier. To solve this problem, we propose a *sample-based collision attack* to infer classification rules.

4.1.1 | Data preparation

As shown in Table 1, we collect phishing websites from PHISHTANK,[34] and legitimate websites from PHISHNET,[35] as subjects to infer classification rules. We only collect URLs of inactive phishing websites. Finally, we collected 3000 phishing websites and 2,566,834 URLs from PHISHTANK and 15,000 legitimate websites with their corresponding webpages from PHISHNET.

Specific data should be extracted from the collected websites to launch collision attacks, as features may differ in classifiers. As aforementioned, we demonstrate our methods using GPPF as the target classifier. GPPF consists of three categories of features: URL-related, Dom-related, and term-related features, as shown in Table 2. URL-related features come from URLs, containing four

TABLE 1 Data set used in this study

Data set types	Number	Source
Phishing URLs	2,566,834	PHISHTANK
Phishing Websites	3000	PHISHTANK
Legitimate URLs	15,000	PHISHNET
Legitimate Websites	15,000	PHISHNET

TABLE 2 Extracted features in GPPF

Feature categories	Feature Types
URL-related	UrlTld=*, UrlDomain=*, UrlOtherHostToken=*, UrlPathToken=*
Dom-related	PageActionURL=*, PageLinkDomain=*
Term-related	PageTerm=*

types of features. Dom-related features come from links and action fields (e.g., form element) in webpages, containing two types of features. Term-related features are collected from texts in webpages, including one type of features (i.e., PageTerm). We extract all links and texts from the collected websites and URLs as the data set for inferring classification rules of GPPF.

4.1.2 | Collision attacks

We demonstrate how to infer classification rules in GPPF by leveraging the collected data set. The main goal is to identify the features used in classification rules. GPPF uses the SHA256 hash algorithm to protect these features. Therefore, we compute the hash value of each feature in the data set using the SHA256 hash algorithm. We then compare its hash value with the hard-coded hash values in the source code of GPPF. If it matches one of the hard-coded hash values in GPPF, then the feature is identified. It remains to understand the semantics of the identified features, which is required to launch white- and gray-box attacks. However, it is difficult to recognize it from the feature names. For example, it is hard to guess the semantics of the feature PageLinkDomain=*, which actually means that "a" element with the "href" attribute whose value is string (i.e., * denotes wildcard). To address this problem, we manually extract the semantics of each feature from the code of GPPF. The results are shown in Table 4, which will be explained in detail in Section 4.4. *This approach is not limited to the* SHA256 *hash algorithm or* GPPF*, instead, we believe it could be used for other hash algorithms and client-side classifiers.*

4.2 | Mutation mechanisms

We propose three mutation mechanisms for white-, gray-, and black-box evasion attacks, which craft adversarial samples by leveraging mutation techniques. During mutation, we always preserve the appearance and functionalities of the websites. This is achieved by carefully

designing mutation operations for implementing our mutation mechanisms. Since changing URLs of websites is very easy for an attacker in real scenarios, but it is difficult to automatically craft adversarial samples via mutating URLs to maintain similarity to the victim's URLs. Therefore, we do not try to modify URLs in our evasion attacks.

4.2.1 | White-box attacks

To evade a classifier with its full knowledge, we should mutate a sample starting from a phishing website by deleting or adding classification rules until the decision score is less than the threshold. The challenge is how to choose such classification rules. Since classification rules can contribute to the decision score either positively or negatively, we partition the classification rules into two sets:

- $R^+ = \{r \in R | w_r > 0\}$ the set of *positive* classification rules indicating when being added the decision score will increase.
- $R^- = \{r \in R | w_r < 0\}$ the set of *negative* classification rules indicating when being added the decision score will decrease.

We address the challenge by deleting positive classification rules and add negative classification rules so that the decision score finally becomes less than the threshold. We leverage a greedy algorithm to choose which classification rules to add or delete aiming at maximally decreasing the decision score at each mutation step. It is worth noting that once a feature is deleted, all the classification rules relying on this feature are also deleted. On the other hand, to add a classification rule, all the features relied on this classification rule should be added. Therefore, we regard each feature of negative classification rules as a mutation "atom" during deletion operation, and regard each positive classification rule as a mutation "atom" during addition operation instead of individual features. At each mutation step, features and classification rules are selected based on their influence on the decision score.

For each feature f that appears in the website p, we define the influence $\delta(p, f)$ of f as

$$\delta(p, f) \triangleq \sum_{r \in R(f) \cap p \vDash r} \left(w_r \cdot \prod_{f' \in F_r} v_{f'} \right),$$

where

- $R(f)$ denotes the set of classification rules that rely upon the feature f,
- for every classification rule $r \in R(f)$ that relies upon the feature f, F_r denotes the set of features that the classification rule r rely upon,
- for every feature $f' \in F_r$ that the classification rule r relies upon, $v_{f'}$ denotes the value of feature f' in the webpage p.

Intuitively, $\delta(p, f)$ is the diff of weights after deleting the feature f.

To add one negative classification rule r, all the features of r should be added, by which more than one classification rules may be added, hence adding the features of one classification

rule may result in the addition of more than one classification rule. Therefore, we define the influence $\delta(p, r)$ of a classification rule r as

$$\delta(p, r) \triangleq \sum_{r' \in R \wedge p \not\models r' \wedge F_{r'} \subseteq F_r} \left(w_{r'} \cdot \prod_{f' \in F_{r'}} v_{f'} \right),$$

indicating the diff of weights after adding all the features of the classification rule r.

Now, we describe the procedure of our white-box attack. For a given phishing website p, we iteratively mutate p as follows until its decision score becomes less than the threshold:

1. if there exists a feature f of a positive classification rule such that $\delta(p, f) > 0, \delta(p, f) \geq \delta(p, f')$ for all other features f' of positive classification rules, and $\delta(p, f) \geq |\delta(p, r)|$ for all negative classification rules r, then delete f from the website p; otherwise goto step 2.
2. if there is a negative classification rule r in the website p such that $\delta(p, r) < 0$ and $\delta(p, r) \leq \delta(p, r')$ for all other negative classification rules r', then add the classification rule r (i.e., all the features of r) into p; otherwise terminate.

4.2.2 | Gray-box attacks

In gray-box scenario, the attacker knows classification rules, but not weights of the classification rules. Therefore, it is impossible to choose the best features or classification rules so that the decision score can be reduced maximally at each mutation step. For efficiency consideration, based on our insights of ML-based classifiers (cf. Section 6.2), we first iteratively delete features that can reduce the decision score; if the decision score is still no less than the threshold after deleting all such features, we add other classification rules of the target classifier that do not exist in the website and check which classification rule can reduce the decision score until no more classification rules can be added. We stop deleting/adding once the decision score becomes less than the threshold. Since we do not know the weights of rules, feature deletion and addition operations may be interleaved during mutation.

4.2.3 | Black-box attacks

In black-box scenario, the attacker neither knows classification rules nor their weights. Hence, it is impossible to directly delete/add classification rules or features during mutation. To overcome this problem, we iteratively add and/or modify DOM nodes, instead of classification rules or features. According to our insight of existing classification rules in ML-based classifiers, we find that: (1) deleting one feature leads to the deletion of all rules that rely on this feature, and (2) to add a rule, all the features of this rule should be added. Therefore, we first iteratively modify DOM nodes. If the decision score becomes less than the threshold, we stop mutating. Otherwise, we iteratively and randomly add nodes that are extracted from legitimate websites but do not exist in the phishing website, to reduce the decision score. If the decision score becomes smaller, we continue mutating the current website. Otherwise, we continue mutating the previous one. We will elaborate details in the next subsection. All these operations are carefully designed to preserve functionality and appearance of the phishing webpage, but without using the obtained features and weights.

4.3 | Mutation operations

According to the above mechanisms, our white- and gray-box attacks need to delete and/or add features. These operations are achieved by manipulating DOM nodes, the same as in black-box scenario. We extract the DOM nodes related to a feature, and add/delete this feature by manipulating these nodes. For example, consider the feature named `PageHasPswdInputs` in `GPPF`, here we assume that "`PageHasPswdInputs` = True" in the sample website. To delete this feature, we identify the nodes (i.e., `input`) related to this feature, then modify these nodes to break "`PageHasPswdInputs` = True." Hence, manipulating DOM nodes is a key operation to achieve our evasion attacks. However, directly manipulating DOM nodes will destruct appearance and/or functionalities of webpages, which are determined by the DOM tree and their nodes. To address this challenge, we introduce two mutation operations: node "addition" and node "modification," which respectively adds and modifies DOM nodes so that (G1) the values of features in positive classification rules become zero, or (G2) the values of features in negative classification rules become positive, but without destructing the appearance and functionalities.

4.3.1 | Node modification

To achieve G1 or G2, we illustrate node modification for different types of nodes (i.e., attribute nodes and text nodes) as follows.

Attribute nodes

Attribute nodes are modified by replacing them with other grammars so that these nodes will not be the features of the classifier. For instance, we can use "event" in `HTML` to achieve this goal. `HTML4` and `HTML5` support the event trigger actions in a browser, and the effect is the same as executing a script when a user clicks on an element. In addition, to ensure the same appearance, we also need to add the style attribute. For instance, the attribute "type" whose value is "submit" in the button element can be modified, as shown below.

```
<style type="text/css">button[type=submit]{height:38px}</style>
<button type="submit"></button> //original
```

We replace the attribute "type" with value "submit" by leveraging the "onclick" event, parse the "css" file and "style" tags to find the styles defined for the button, and add these styles to the button so that the appearance of the button remain unchanged. The crafted button is shown below.

```
<button style='height:38px;'onclick="this.type='submit'";></button> //crafted
```

It is very difficult for the users to perceive such changes since the two actions do not destruct the appearance and functionality of webpages. Table 3 lists the function-related attribute nodes which can be modified. We remark that not all attribute nodes can be modified, as some of them are related to webpage appearance (e.g., "align" attribute in a "p" element and "background" attribute in a "body" element) unless the appearance can be imitated by changing "style" attribute. For example, we cannot modify the attribute "type" whose value is "radio" in "input" element, because its appearance cannot be imitated by changing "style" attribute.

TABLE 3 Modifiable function-related attributes where * refers to the "type" attribute in "input" element, which can be deleted only when its value is "reset," "button," or "password"

Element	Function-related Attributes	Event
button	form, formaction, formenctype, formmethod, formnovalidate, formtarget, name, type, value	onclick
input	accept, submit, text, type*, step, size, required, name, placeholder, min, max, formtarget, formnovalidate, formmethod, formenctype, formaction, form, accept	onfocus
form	action, enctype, method, name, novalidate, target	oninput
a	download, href, hreflang, media, rel, target, type	onclick
table	summary	onmousemove

Text nodes

Classifiers usually check text through string comparison. Thus, we can modify text nodes by inserting an invisible (e.g., the zero-width space "") character, which looks like that it does not exist. For example, a text node "Hello World!" in a "p" element, as shown below

```
<p>Hello World!</p> //original
```

It can be modified by inserting the character "" between "l" and "o," displayed the same as the original one, resulting in the following code:

```
<p>Hell&#65279;o World!</p> // crafted
```

4.3.2 | Node addition

To preserve the appearance and functionalities of webpages, we demonstrate node addition by adding *invisible* element nodes, that is, with no size and color. We remark that the parent of an attribute or text node is an element node. If the element node is invisible, then all its children are also invisible. Therefore, to add attribute and text nodes, we add them as children of an invisible element node.

However, in black-box scenario, we do not know any information about the classifier and thus cannot determine which node to add to hit negative classification rules after node modification. We assume that there are more negative classification rules in legitimate websites than in phishing ones. Therefore, based on this assumption, we collect three types of nodes (i.e., element nodes, attribute nodes, and text nodes) from the collected legitimate websites. After that, we randomly select nodes from the collected data and add them to the target website. To avoid adding useless nodes in black-box scenario, we come up with a *rollback mechanism*. Specifically, if the decision score does not decrease after adding a specified number (e.g., 3) of nodes, we roll back to the previous sample and continue to add other nodes. Otherwise, we continue adding nodes into the latest one.

In summary, all nodes can be added into the webpages, but only attribute and text nodes can be modified according to mutation operations.

Though our node modification and addition are simple, they are effective in our experiments. It might be able to improve the resilience of classifiers by proposing new classification rules following disclosure of our findings, we believe more sophisticated methods (e.g., script and Shadow DOM[36]) could be leveraged to achieve node modification and addition, which are widely used in legitimate websites, but very difficult to analyze statically.

4.1 | Concretizing our attacks on GPPF

In this section, we show how to concretize our attacks on the classifier `GPPF`. We successfully extracted the 2130 classification rules from the source code of `Chromium`, in which each classification rule r consists of a classification rule identity (ID), a weight w_r and a set F_r of hashed features. In total, there are 1009 hashed features. Table 4 lists all features excluding URL-related features in `GPPF`. As aforementioned, all the features can be added, while only some features can be deleted. For example, the feature `PageHasForms` denotes whether a webpage has a "form" node. The "form" node is an element node that cannot be modified. We cannot delete features 4 and 5 neither, as they are appearance-related attribute nodes. Features 6, 7, 8, and 9 representing the frequency of special links can be deleted by adding links whose domain is internal. When the frequency value is lower than a predefined threshold in `GPPF`, `GPPF` cannot detect such features. We cannot delete script-related features 10 and 11, as deleting them may affect the functionalities of websites.

Features 2, 3, 12, and 13 can be deleted by using the attribute nodes (e.g., `event`) which implement the same functionality but making them undetectable by the classifier. For example, feature 3 indicates whether the website has an "input" element whose "type" is "password," as shown below:

```
<style type="text/css">input[type=password]{width:8px;}</style><input type="password"/> // original
```

After modifying it using the `event` attribute node, it becomes

```
<input style="width:8px" onfocus="this.type='password'"/> //crafted
```

Thus, the classifier cannot detect this feature. For feature 14, we can delete it directly since it is a term-related feature.

5 | DEFENSE METHODS

In this section, we first propose a similarity-based defense method, named Pelican, to detect evasion attacks. Pelican addresses the scenario where a phishing website is already detected by a classifier, and the attacker is trying to mutate it to mislead the classifier. We then propose two classification rule selection strategies to increase attack cost, hence enhancing the robustness of the classifier.

5.1 | Similarity-based evading detection

Our detection is based on the following two observations of mutation-based evasion attacks: (1) adding or deleting features do change the structures of the webpages, but, (2) the change is

TABLE 4 Dom-related and Term-related features in GPPF, where "*" refers to wildcard

	Feature Types	Del	Add	Semantics
1	PageHasForms	✗	✓	Whether there is a form element in the webpage
2	PageHasTextInputs	✓	✓	Whether there is an input element whose type is "text"
3	PageHasPswdInputs	✓	✓	Whether there is an input element whose type is "password"
4	PageHasRadioInputs	✗	✓	Whether there is an input element whose type is "radio"
5	PageHasCheckInputs	✗	✓	Whether there is an input element whose type is "checkbox"
6	PageExternalLinksFreq	✓	✓	$\frac{\#ext\_links}{\#total\_links}$, where `#total_links` is the No. of "href" attributes in "a" elements and `ext_links` is the No. of "href" attributes in "a" elements whose domains are external
7	PageActionOtherDomainFreq	✓	✓	$\frac{\#actions_{ext}}{\#total\_actions}$, where `#total_actions` is the number of the "action" attributes in the form element and $\#actions_{ext}$ is the number of "action" attributes whose domains are external
8	PageSecureLinksFreq	✓	✓	$\frac{\#secure\_links}{\#total\_links}$, where `#total_links` is the number of "href" attributes in "a" elements and `#secure_links` is the number of "href" attributes whose scheme is https in "a" elements
9	PageImgOtherDomainFreq	✓	✓	$\frac{\#imgs_{other\_domain}}{\#total\_imgs}$, where `#total_imgs` is the number of "img" elements in the webpage. $\#imgs_{other\_domain}$ is the number of "srr" attributes whose domains are external
10	PageNumScriptTags>1	✗	✓	Whether the number of "script" tag is greater than 1
11	PageNumScriptTags>6	✗	✓	Whether the number of "script" tag is greater than 6
12	PageActionURL=*	✓	✓	Whether there is a form element with the "action" attribute whose value is a string
13	PageLinkDomain=*	✓	✓	Whether there is an "a" element with the external "href" attribute whose value is a string
14	PageTerm=*	✓	✓	Whether there is a text node containing a string

limited. Therefore, if a website has a high "similarity" compared with a recently detected phishing one, we regard it as an adversarial sample crafted by evasion attacks. We remark that our approach aims at detecting evasion attacks rather than general phishing attacks. It can be deployed with a ML-based classifier to prevent it from mutation-based evasion attacks.

5.1.1 | DOM tree similarity (baseline)

We first propose to compute the similarity of DOM trees as a baseline. We denote by T and T' two DOM trees of a recently detected phishing website and an unknown website. If the similarity between T and T' is high, we conclude that T' is mutated from T, thus successfully identify the crafted phishing website without using the ML-based classifiers.

In detail, we traverse the DOM trees T and T' using a breadth-first search algorithm and compute the hash values of the element nodes in each layer. As there are two types of nodes (i.e., text nodes and attribute nodes) that rely on an element node, thus, to measure the similarity of two element nodes, we compare both the hash values of their text and attribute nodes. We only consider the similarity of two elements with the same tag name. If two elements have distinct tag names, they are regarded as different elements. The similarity of elements in the i^{th} layer is defined as:

$$\texttt{similarity}_{\texttt{ele}}(\texttt{E}_\texttt{i}, \texttt{E}'_\texttt{i}) \triangleq \frac{|\texttt{a}_1^\texttt{i} \cap \texttt{a}_2^\texttt{i}|}{|\texttt{a}_1^\texttt{i} \cup \texttt{a}_2^\texttt{i}|} + \frac{|\texttt{t}_1^\texttt{i} \cap \texttt{t}_2^\texttt{i}|}{|\texttt{t}_1^\texttt{i} \cup \texttt{t}_2^\texttt{i}|}$$

where a_1^i (resp. a_2^i) denotes the set of attribute nodes belonging to the element node E_i (resp. E_i'), t_1^i (resp. t_2^i) denotes the set of text nodes belonging to the element node E_i (resp. E_i'). Finally, we compute the proportion of the common nodes $\texttt{comm}_i$ with the same hash values against all the nodes of the current layer i in two DOM trees. The similarity $\texttt{similarity}_{\texttt{tree}}(\texttt{T}, \texttt{T}')$ between T and T' is defined as the average common nodes of each layer, that is,

$$\texttt{similarity}_{\texttt{tree}}(\texttt{T}, \texttt{T}') \triangleq \sum_{i=1}^{m} \frac{\ell_i}{m}$$

where m is the maximal number of layers in the DOM trees, $\ell_i \triangleq \frac{\texttt{comm}_\texttt{i}}{|\, n_1^i \cup n_2^i \,|}$ and $\texttt{comm}_\texttt{i} \triangleq \sum_{j=1}^{|n_1^i \cup n_2^i|} \texttt{similarity}_{\texttt{ele}}(\texttt{E}_\texttt{j}, \texttt{E}'_\texttt{j})$, n_1^i (resp. n_2^i) denotes the set of element nodes at the i^{th} layer of T (resp. T').

5.1.2 | Personalized similarity (Pelican)

The DOM tree similarity method (baseline) is effective to detect most crafted samples, but failed on several cases, for example, adversarial samples crafted by the gray-box attack. After an in-depth analysis, we found there are four types of features related to "frequency," namely, features at No. 6–9 in Table 4. To evade classifiers, it requires to add or modify a large number of nodes, which may significantly reduce the DOM tree similarity, occurring in our gray-box scenario. In addition, if the attacker obtains the knowledge of our DOM tree similarity method, (s)he may add more irrelevant nodes or attributes, invisible elements, and layers to reduce the similarity. However, it is difficult to directly remove the nodes and attributes without

destructing the appearance and functionalities. Therefore, we propose a *personalized similarity method*, named Pelican, that is defined as follows:

$$\texttt{similarity}_{\texttt{ele\_Pelican}}(\texttt{E}_\texttt{i}, \texttt{E}'_\texttt{i}) \triangleq \frac{|\texttt{a}_1^\texttt{i} \cap \texttt{a}_2^\texttt{i}|}{|\texttt{a}_1^\texttt{i}|} + \frac{|\texttt{t}_1^\texttt{i} \cap \texttt{t}_2^\texttt{i}|}{|\texttt{t}_1^\texttt{i}|},$$

$$\texttt{similarity}_{\texttt{Pelican}}(\texttt{T}, \texttt{T}') \triangleq \sum_{\texttt{i}=1}^{\texttt{m}} \frac{\ell_\texttt{i}}{\texttt{m}}, \ell_i \triangleq \frac{\texttt{comm}_\texttt{i}}{|n_1^i|} \quad \text{and} \quad \texttt{comm}_\texttt{i} \triangleq \sum_{j=1}^{|n_1^i|} \texttt{similarity}_{\texttt{ele\_Pelican}}(\texttt{E}_\texttt{j}, \texttt{E}'_\texttt{j})$$

where $\texttt{similarity}_{\texttt{ele\_Pelican}}(\texttt{E}_\texttt{i}, \texttt{E}'_\texttt{i})$ is the similarity of elements in the i^{th} layer by using Pelican, m is the number of layers in the DOM tree T, and n^i denotes the set of element nodes at the i^{th} layer in T. In particular, to defend against attacks that insert layers into the DOM tree during mutation, Pelican continues comparing the next layer of T' until the two layers are similar. If all the remaining layers are not similar enough, we just roll back to compute the similarity of elements in the same layer.

5.1.3 | Application scenario in practice

Figure 3 depicts the application scenario of our defense method Pelican in practice. For each website, the system first queries whitelist and blacklist to quickly identify legitimate and known phishing website in a standard way. Whitelist and blacklist can be customized by users or use third-party resources.[9,34,35] If the website is neither in whitelist nor blacklist, then Pelican is applied to check whether the website is crafted by an evasion attack. If it is an evasion attack, a phishing warn alerts without passing to the ML-based classifier. Otherwise, the ML-based classifier is used to check the website. The website is added into the phishing website database for Pelican if it is classified as a phishing one. To avoid storing all the detected phishing websites, we only need to store either recent k phishing websites and/or the phishing websites detected in recent h hours. This is feasible in practice, as browsers also cache recently visited webpages in disks. By doing so, Pelican can prevent from querying directly and efficiently to the ML-based classifiers by the attackers, but still detect phishing attacks.

5.2 | Classification rule selection

To enhance the robustness of the classifiers, we introduce two classification rule selection strategies to increase attack cost.

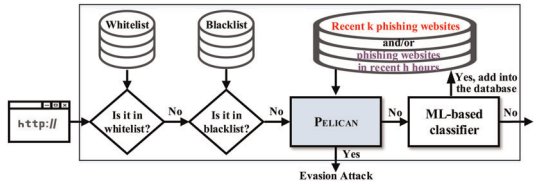

FIGURE 3 Application scenario in practice [Color figure can be viewed at wileyonlinelibrary.com]

5.2.1 | Negative subset classification rule pruning

Given two classification rules r and r', we say r' is a *sub-rule* of r, if the set of features of r subsumes the one of r'. Such subset classification rules decrease the attack overhead, because if the attacker has inferred the classification rule r', then the classification rule r as well. Attackers can improve the attack effectiveness and efficiency by leveraging negative subset classification rules. When the attacker adds all the features of a negative classification rule r into a website, both classification rules r and r' will be added into the sample. Therefore, the mutation can maximally reduce the decision score by adding all features of r. To enhance the robustness of the classifiers, it is better to prune subset classification rules, which is demonstrated by our experiments (cf. Section 6.4.2).

5.2.2 | Single classification rule pruning

A classification rule is *single* if it relies on features that are not relied on any other classification rules. Single classification rules are completely independent of other classification rules, deleting or adding them will not affect any other classification rules. Therefore, attackers can aggressively add all negative single classification rules if they are addable, which can definitely reduce the decision score. In addition, in white-box scenario, attackers can also directly delete all positive single classification rules. Addition and deletion of single classification rules are very efficient and do not have any side effect on other classification rules. Therefore, single classification rules have a potentially severe hazard to classifiers and should be pruned to enhance the robustness of the classifiers. This also has been confirmed by our experiments (cf. Section 6.4.3).

6 | EXPERIMENTS

In this section, we first introduce the data set and experimental settings, then demonstrate our sample-based collision attack for inferring classification rules, and finally, evaluate our proposed evasion attacks and defense methods for evading detection and robustness improvement of the classifier.

6.1 | Data set and experimental settings

6.1.1 | Data set

We use two data sets (cf. Table 1), for evaluating our collision attack. We tested 3000 phishing websites using the classifier `GPPF`. Among them, 305 can be detected by `GPPF` (i.e., $score \geq 0.5$). Remark that this does mean that `GPPF` is not powerful for detecting phishing websites, as the other undetected phishing websites are inactive and out of date. We use these 305 phishing websites as the subjects to evaluate evasion attacks and defense methods.

6.1.2 | Settings

As mentioned in Section 1, we examined several academic and industrial ML-based tools, all the other tools (except GPPF) either are not publicly accessible or cannot output decision scores for each input which is required even in our black-box scenario. Thus, we demonstrate our attacks on the state-of-the-art ML-based classifier GPPF.

We use DEPOT_TOOLS[37] to obtain the code of Chromium (version: 72.0.3626), and use ninja[38] to compile Chromium. We target the classifier GPPF which is deployed in Chromium. To obtain the decision score of each website, we added an instruction that prints the return value of the function ComputeScore in the file scorer.cc. We run Chromium in UI mode for our experiments. Our experiments are carried out on a server (i.e., Ubuntu 16.40 or Windows 10) with 32 GB Memory and 1 TB hard disk.

To demonstrate that our black-box attack is generic and effective for other practical ML-based classifiers, we launch transferability attacks on the industrial tool TrafficLight, coming from BitDefender, a popular security company. It is able to detect phishing webpages and part of the processing is done in the cloud with some intelligent engines. Recall that BitDefender TrafficLight is proprietary without any publicly available information about the internal design and implementations, hence completely black-box. However, only 32 out of 305 seeds can be detected by BitDefender TrafficLight. Thus, we use all the crafted adversarial samples (32×3) from these 32 phishing websites as subjects. We also managed to launch transferability attacks on the industrial tools: Netcraft Anti-Phishing Extension and 360 Internet Protection, as add-ons of Chrome. However, both tools failed to detect any of the 305 seeds. We guess this may be due to that the phishing websites are deployed in a controllable network with a fake URL, while Netcraft Anti-Phishing Extension and 360 Internet Protection detect phishing websites using URL.

6.2 | Evaluation of collision attack

To evaluate the effectiveness of our *sample-based collision attack*, we demonstrate on the classifier GPPF. We leverage our collected phishing and legitimate data set from PHISHTANK and PHISHNET in Table 1. We directly recovered 14 hashed features out of 1009 features after analyzing Chromium, while the remaining 995 hashed features cannot be directly recovered. Our collision attack is able to successfully infer 878 features (88.24%) out of 995 hashed features, as shown in Table 5.

In detail, our collision attack successfully inferred 491 URL-related and Dom-related features in 0.68 h, and 387 term-related features in 11.66 h. Term-related features include (1) containing numbers, (2) involving Arabic, or (3) using the degree as a value such as "1°." *For*

TABLE 5 Inferred features

Feature categories	#Total features	#Inferred features	Time (h)
URL-related and Dom-related	563	491	0.68
Term-related	432	387	11.66
Total	995	878	12.34

ethical consideration, we do not provide details of concrete features. Our collision attack successfully inferred them because those terms are easy to be found in real-world websites.

Recall that each classification rule r in GPPF consists of a classification rule identity (ID), a weight w_r and a set F_r of hashed features, where the weights of classification rules are given in plaintext in the source code of Chromium. Using successfully inferred hashed features, we fully inferred 1,952 (91.64% = 1952/2130) classification rules in total based on the inferred features. However, not all of them can be used for evasion attacks because some of them are related to appearance or functionalities or URLs. Table 6 shows the 1327 classification rules that can be used for our evasion attacks, in which 427 classification rules are negative classification rules that can be used for addition, and 941 classification rules are positive classification rules that can be used for deletion. We remark that it is nontrivial to fully resolve all the features of these inferred classification rules, as we can only know partial features of some classification rules. Such classification rules can only be deleted during evasion attacking. Therefore, classification rule deletion is considered to be more effective than classification rule addition.

6.3 | Evaluation of evasion attacks

6.3.1 | Effectiveness and efficiency

To evaluate the effectiveness and efficiency of our evasion attacks, we respectively conduct white-, gray- and black-box attacks against the classifier GPPF using 305 phishing websites as seeds. The results of our proposed evasion attacks are shown in Table 7. Column "Score" shows the decision score range, with the corresponding number of phishing websites in Column "#Phishing Websites." Columns "#Succeeded Websites" indicate the number of crafted adversarial websites. Columns "Mutated Time" indicate the average time spent on crafting the adversarial websites. Columns "#Mutated Features/Rules" show the average number of mutation operations (i.e., deletion and addition) for crafting the adversarial websites, which also indicates query times to the classifier. Remark that the number of mutated classification rules is always larger than that of features, as when one feature is mutated, classification rules that rely on this feature are also mutated.

From the results in Table 7, we can see that our white- and gray-box attacks achieve 100% attack success rate, and our black-box attack successfully crafts 298 adversarial samples by only

TABLE 6 Statistic of inferred classification rules

Rule weight	#Total	#Addable	Rule weight	#Total	#Deletable
[−6.0, −7.0)	2	1	[6.0, 7.0)	6	4
[−5.0, −6.0)	2	0	[5.0, 6.0)	13	8
[−4.0, −5.0)	6	3	[4.0, 5.0)	39	26
[−3.0, −4.0)	30	15	[3.0, 4.0)	114	76
[−2.0, −3.0)	110	48	[2.0, 3.0)	266	175
[−1.0,−2.0)	298	157	[1.0, 2.0)	455	307
[0.0, −1.0)	359	203	[0.0, 1.0)	430	345
Total	807	427		1323	941

TABLE 7 Performance of our attacks in white-, gray- and black-box scenarios

Score	#Phishing websites	#Succeeded websites	Mutation time (ms)	#Mutated features/rules
Performance of white-box attacks				
1	1	1	572.00	5.00/30.00
[0.9,1.0)	131	131	91.00	1.65/9.81
[0.8,0.9)	59	59	49.59	1.10/4.51
[0.7,0.8)	14	14	48.71	1.00/3.93
[0.6,0.7)	68	68	26.35	1.00/4.32
[0.5,0.6)	32	32	32.22	1.00/2.31
Total	305	305	–	–
Performance of gray-box attacks				
1	1	1	966.00	14.00/48.00
[0.9,1.0)	131	131	165.76	6.81/19.77
[0.8,0.9)	59	59	121.63	6.22/16.14
[0.7,0.8)	14	14	194.86	5.50/12.57
[0.6,0.7)	68	68	139.25	5.01/12.53
[0.5,0.6)	32	32	106.38	3.84/7.50
Total	305	305	–	–
Performance of black-box attacks (node modification only)				
1	1	1	606.32	12.00/50.00
[0.9,1.0)	131	124	244.96	4.27/20.70
[0.8,0.9)	59	59	206.34	1.97/9.76
[0.7,0.8)	14	14	291.00	1.93/8.36
[0.6,0.7)	68	68	231.97	2.29/0.03
[0.5,0.6)	32	32	219.94	1.09/4.69
Total	305	298	–	–

modifying DOM nodes for 305 seeds. (Note that in the black-box scenario, we first modify nodes and then add nodes, while node modification and addition operations may be mixed during mutation in white- and gray-box scenarios. Although features are deleted first in gray-box attacks, the feature deletion is achieved by node modification and addition.)

The black-box attack failed to craft adversarial samples for seven websites (cf. Table 8) by only modifying nodes. Therefore, we continue mutating them by adding nodes. Table 8 shows the results, which target five domains. Column "Score after modification" gives the decision score after node modification. Column "#Addition Operations" gives the number of node addition operations. Column "Mutate Time" gives the time of node addition operations.

From the results in Table 8, we can observe that the adversarial samples of all the 7 websites can be crafted by node addition, using at least 543 steps. After a deep analysis of these added nodes, we found that only one or two classification rules are changed. This demonstrates that node

TABLE 8 Seven phishing websites that fail to evade GPPF by only modifying nodes in black-box scenario

Target domain	Score after modification	#Addition operations	Mutate time (ms)
sso.godaddy.com/login	0.81	543	62.87
passport.alibaba.com/icbu_login.htm	0.83	543	56.12
store.cpanel.net/login/	0.84	543	60.10
store.cpanel.net/login/	0.84	543	57.61
store.cpanel.net/login/	0.84	543	58.27
passport.alibaba.com/icbu_login.htm	0.90	543	60.76
passport.alibaba.com/icbu_login.htm	0.93	1032	299.54

TABLE 9 Classification rules that cannot be deleted in the real case

Rule	Features	Rule weight
1	PageNumScriptTags>6, PageExternalLinksFreq	0.55
2	UrlPathToken=*	2.23
3	PageNumScriptTags>1,PageNumScriptTags>6, PageExternalLinksFreq	0.52
4	UrlPathToken=*, PageHasForms	0.21
5	PageNumScriptTags>6, UrlPathToken=*	0.46
6	PageNumScriptTags>1, UrlPathToken=*	1.17
7	PageNumScriptTags>1, UrlPathToken=*	0.43
8	PageImgOtherDomainFreq	0.34
9	PageNumScriptTags>1, PageExternalLinksFreq	1.48

modification is more efficient than node addition in black-box scenario. The most difficult website is the last one whose initial decision score is 0.99. After modifying all modifiable nodes, its decision score becomes 0.93, which is still larger than the threshold 0.5. It takes 1032 times of addition operations to reduce its decision score below 0.5. Table 9 shows the details of classification rules that are hit by the website, but cannot be deleted. Column "Features" represents the features of each classification rule. Although we delete many positive classification rules from the website, the weights of the remaining classification rules hit by the websites are still larger enough so that the decision score is still larger than 0.5. Though the black-box attack on these seven websites requires a relatively large number of queries, the number of queries on the other 298 websites is comparable with that of white-box attack. We remark that state-of-the-art black-box attacks on image classifiers,[16] PDF malware classifiers,[32] and speaker classifiers[15] also require a much larger number of queries, due to the lack of knowledge of the classifier.

6.3.2 | Visual impact

To evaluate the visual impact of the mutated webpages, we capture screenshots of all webpages of phishing and crafted adversarial websites and compare each phishing screenshot with its

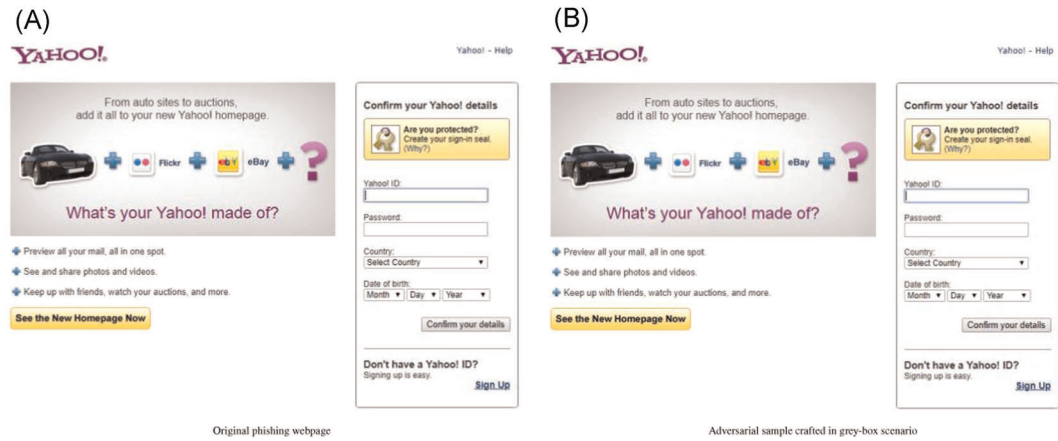

FIGURE 4 Screenshots of two webpages targeting yahoo, (A) Original phishing webpage and (B) Adversarial sample crafted in grey-box scenario [Color figure can be viewed at wileyonlinelibrary.com]

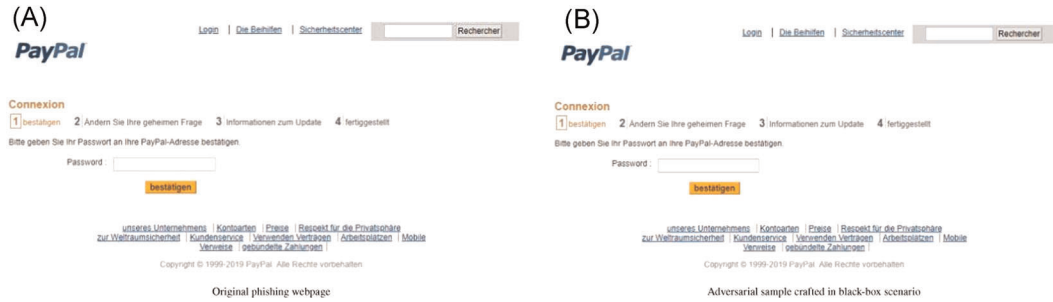

FIGURE 5 Screenshots of two webpages targeting PayPal, (A) Original phishing webpage and (B) Adversarial sample crafted in black-box scenario [Color figure can be viewed at wileyonlinelibrary.com]

adversarial ones using MSE[24] and MAE[25] criterion, two common distortion measures of images. All the values of MAE and MSE are 0, which demonstrates that our attacks preserve appearance. We discuss two cases of visual impact which respectively target Yahoo and PayPal.

Figure 4 shows the screenshots of the original phishing website targeting Yahoo (Figure 4A) and an adversarial sample (Figure 4B), crafted in gray-box scenario. Our tool added 17 `PageTerm` features, one `PageImgOtherDomainFreq` feature, and one `PageHasPswdInputs` feature, deleted one `PageExternalLinkFreq` feature and two `PageTerm` features. Adding `PageImgOtherDomainFreq` and deleting `PageExternalLinkFreq` features are achieved by adding 3321 `<a>` nodes with internal href and 554 `<img>` nodes with external href. The values of both MAE and MSE are 0.

Figure 5 shows the screenshots of a phishing website targeting PayPal (Figure 5A) and an adversarial sample (Figure 5B), crafted in black-box scenario. Our tool deleted three`PageTerm` features, one `PageHasPswInputs` feature, and one `PageLinkDomain` feature. The values of both MAE and MSE are 0.

TABLE 10 Transferability of crafted samples

Attack	#Detected/#Crafted	Detection ratio
White-box	32/32	100%
Gray-box	16/32	50%
Black-box	6/32	18.75%

Transferability attack

As aforementioned, we evaluate the transferability attack on BitDefender TrafficLight. Table 10 shows the results of the transferability attack on BitDefender TrafficLight. We can observe that adversarial samples crafted by the black- and gray-box attacks achieved 81.25% and 50% transferability attack rate. On contrary, BitDefender TrafficLight detected all the adversarial samples crafted by the white-box attacks. This is because our white-box attack only made minor changes that are very specific to features/rules of GPPF, whereas black-box attack made more changes independent of GPPF, hence, is more robust than others for transferability attacks.

As aforementioned, BitDefender TrafficLight is completely black-box and part of its processing is done in the cloud with some intelligent engines, therefore it is impossible to find the reason why adversarial examples crafted by our attacks on GPPF are still effective against BitDefender TrafficLight. One possible reason is that BitDefender TrafficLight may leverage similar features as GPPF to decide whether a website is phishing or not.

6.4 | Evaluation of defense methods

6.4.1 | Evaluation of similarity-based evasion detection

We evaluate both DOM tree similarity method and Pelican against three evasion attacks. For each method, we compare the similarity between the adversarial sample and its original phishing website. We use the 305 original phishing websites and their corresponding adversarial samples as the subjects. Figure 6 shows the results of similarity comparison using two methods under three attacks. We can observe that both the DOM tree similarity method and Pelican achieve high similarity (i.e., over 90%) on most of the adversarial samples crafted by white- and black-box attacks. Surprisingly, on the adversarial websites crafted by gray-box

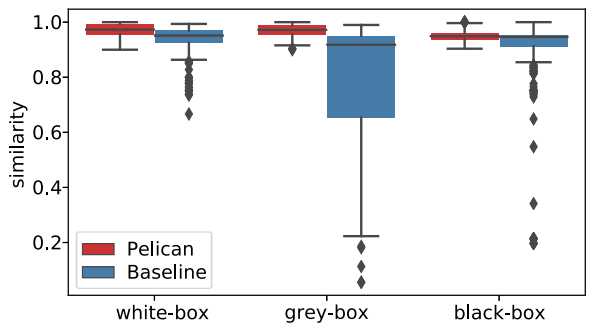

FIGURE 6 Results of similarity comparison [Color figure can be viewed at wileyonlinelibrary.com]

attacks, Pelican significantly outperforms the DOM tree similarity method. After a further investigation of these websites, we find that there are four frequency-related features in GPPF, adding or deleting these features needs to mutate a lot of nodes to reduce/increase the frequency of some elements. Hence, the mutation of these features adds lots of invisible nodes, which reduces DOM tree similarity by increasing the number of nodes.

For example, a phishing website with domain "login.microsoftonline.com.best10reviews.com" has decision score 0.92. It hits a classification rule with weight 2.00 in the classifier, the classification rule consists of two features: "PageSecureLinksFreq" and "PageHasPswdInputs." "PageSecureLinksFreq" is a frequency-related feature, computed as $\frac{\#secure\_links}{\#total\_links}$. Currently, it equals to 1, indicating that all the "href" links in the website are secure (i.e., "https"). To delete this feature, our gray-box attack adds lots of insecure (i.e., "http") and invisible "href" links. Consequently, the DOM tree similarity is 46.78%, thus can bypass the DOM tree similarity-based detection. However, such frequency-related features have no effect on Pelican, as Pelican computes the similarity of the number of nodes in the phishing website, instead of that in the crafted sample. Thus, such websites can be detected by Pelican.

6.4.2 | Evaluation of subset classification rule pruning

We found that GPPF has at least 150 addable negative subset classification rules. Therefore, we conduct experiments on GPPF without subset classification rules to investigate the impact on attack cost and accuracy. To remove subset classification rules in GPPF, we set the values of such classification rules to 0, as it is impossible for us to retrain GPPF.

Impact on attack cost

We automatically create 50 personalized phishing websites (10 websites in each decision score range) as our subjects. Inspired by the seven phishing websites in Table 8 on which feature deletion fails to craft adversarial examples in black-box scenario, we first collect static webpages from legitimate websites and change the action field of HTML forms in these websites for phishing, then delete all deletable features, and finally add some undeletable features to make their decision scores within the desired range.

We compare the attack costs with and without using subset classification rules in black-box scenario. Table 11 shows the results. We can observe that the attack requires more operations to craft an adversarial sample in black-box scenario without using subset classification rules, hence, conclude that pruning subset classification rules can increase the attack cost.

TABLE 11 Attack cost comparison with/without subset classification rules

Score	#Websites	#Operations (with)	#Operations (without)
[0.9,1.0)	10	1331.7	3015.0
[0.8,0.9)	10	643.4	1028.8
[0.7,0.8)	10	418.2	967.0
[0.6,0.7)	10	273.5	353.8
[0.5,0.6)	10	212.3	221.0

Impact on accuracy

We check all 305 original phishing websites and 15,000 legitimate website using GPPF without subset classification rules. Although, there are 2017 legitimate and 232 phishing websites whose decision scores changed after disabling subset classification rules, the changes are small so that none of them are misclassified by the classifier.

6.4.3 | Evaluation of single classification rule pruning

We found that GPPF has at least 34 addable and 39 deletable single classification rules, as shown in Table 12. Therefore, we conduct experiments on GPPF.

Impact on attack cost

To understand the impact on attack cost, we conduct experiments in white-box and gray-box scenarios by only using single classification rules for mutation. We achieved a 100% attack success rate using only these single classification rules. An adversarial sample is crafted in less than 1ms for the 305 phishing websites. Therefore, we conclude that single classification rules have a potentially severe hazard to classifiers, and should be pruned from the classification rules to enhance the robustness.

Impact on accuracy

To understand the impact on accuracy, we set the values of all deletable or addable single classification rules to 0 for all websites. We check all 305 original phishing websites and 15,000 legitimate websites using GPPF. Although there are 60 legitimate and 29 phishing websites whose decision score changed after disabling single classification rules, none of them is misclassified by GPPF.

7 | RELATED WORK

7.1 | Phishing website detection

Phishing website detection has been studied in several work, such as blacklist,[3] similarity-based,[4] and ML-based[5,9] anti-phishing solutions. The blacklist approach maintains a list of URLs of known phishing websites, but it is only able to detect phishing websites whose URLs are in the list. Similarity-based approaches compare similarity between authentic and phishing webpages, differing in terms of: features extracted to identify similarity and the matching algorithm used. Similarity-based approaches are able to detect new phishing websites, although not greatly improve the accuracy of other existing approaches.[20] ML-based approaches train a

TABLE 12 Single classification rules in GPPF

Feature	#Deletable	Total weight	#Addable	Total weight
PageTerm=*	23	43.42	14	−22.94
PageLinkDomain=*	16	39.96	20	−19.89
Total	39	83.38	34	−42.83

classification model using features/rules from known phishing websites and authentic websites, differing in terms of: features for training and ML algorithm used. They are scalable and accurate.

However, blacklist-based classifiers are shown vulnerable to evasion attacks by Oest et al.[3] In this study, we showed that ML-based classifiers are also vulnerable to evasion attacks even in black-box scenario. We conjecture that classifiers based on other approaches are also vulnerable to evasion attacks, in particular in white- and gray-box scenarios, for which the adversary can gain details of classifiers. The similarity-based approaches are closely related to our defense method Pelican. The main differences are: (1) Pelican aims at detecting evasion attacks rather than general phishing websites; (2) Pelican compares the similarity between an unknown one and recently detected phishing webpages rather than the similarity between an unknown one and authentic webpages; and (3) we compute similarity layer-by-layer and only need to record recently detected phishing webpages due to (1), hence is both time- and space-efficient.

7.2 | Attacks on classifiers

Evasion attacks have been exhibited in different scenarios such as malicious PDF files[12,32] and malware.[13,39] However, domain-specific characteristics that usually pose new challenges should be taken into account in different scenarios. For example, adversarial images usually minimize distortion when adding pixel-level noise,[40] and adversarial (PDF) malware usually need to keep original malicious functionalities[13] *only*. However, adversarial phishing websites usually have to preserve both appearance and functionalities which arise new challenges compared with other scenarios. These new challenges potentially make this study area less noticed and more difficult. For instance, although the method proposed by Xu et al.[32] is generic for crafting adversarial samples. This method leverages an evolutionary algorithm to mutate PDF malware and relies on an oracle to determine if a generated variant preserves maliciousness. It is nontrivial to be directly adapted into our setting. In addition, in recent years, attacks on deep neutral network based-classifiers are also widely studied[40], which have the similar theory by leveraging adversarial images in ML. In summary, all these existing studies consider domain-specific scenarios with many differences compared with ours in phishing website detection which owns new challenges.[41] proposes a white-box attack to evade phishing website detection. However, this approach not only has access to the information of the detector, but also destructs the appearance and functionalities of webpages, making them less effective for web phishing.

7.3 | Defenses for classifiers

It is not surprising that many defense solutions have been proposed to enhance the robustness of ML-based classifiers. Saeys et al.[42] use ensemble feature selection techniques and demonstrate that multiple feature selection methods can receive more robust results. Biggio et al.[43] use multiple classifiers to gain a better result in adversarial environments. Goodfellow et al.[44] retrain on adversarial samples to improve the robustness of classifiers, and highlight those ensemble methods have some limitations for evasion attacks. Zhang et al.[45] use adversarial feature selection to increase the robustness of classifiers. In the area of malware, Chen et al.[13] propose a similarity-based approach for filtering Android crafted samples by leveraging the

most benign and malicious samples. In addition, many recent studies focus on identifying adversarial samples in deep neural networks.[46,47] These solutions are all preventive in advance. However, all these studies are tailored to a different setting than ours. It is difficult to apply these solutions to enhance the robustness of the phishing website classifiers directly due to the domain-specific characteristics.

As for the client-side classifiers, the effective solution is to increase the complexity of reverse engineering by using more advanced obfuscation techniques. The less information an attacker gets from a classifier, the safer the classifier is. In terms of the defense solutions for phishing website classifiers with evasion attacks, to our best knowledge, there are no existing studies that focus on it.

8 | CONCLUSION AND FUTURE WORK

In this study, we conducted the first systematic study on evasion attacks of ML-based web phishing classifiers and revealed their weaknesses. Specifically, we proposed a novel sample-based collision attack which is able to effectively and efficiently infer almost all the hashed features and classification rules of the classifier `GPPF`. We also proposed a class of advanced evasion attacks for three attack scenarios (i.e., white-, gray-, and black-box) by leveraging mutation techniques and the first defense method against evasion attacks. We further presented two classification rule selection strategies to enhance the robustness of the classifiers and increase the attack cost. Experimental results demonstrated that our attacks can achieve 100% attack success rate in less than one second per seed on `GPPF`, and 81.25% transferability attack rate on `TrafficLight`. Our defense method is effective in detecting crafted adversarial websites, hence can be used to detect evasion attacks and enhance the robustness of classifiers. Our study shows that a sophisticated phishing attack is still a severe cyber security vector.

Currently, our attack mutates seeds by leveraging a greedy algorithm. In the future, we plan to investigate attacks by leveraging genetic algorithms, which have been used to fool PDF malware classifiers[32] and face recognition systems.[48] It is interesting to understand which algorithm is more effective and efficient. Another future direction is to investigate attacks against ML-based classifiers that only outputs the decision result instead of the decision score for each input test.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China (Grant No. 62072309), the National Research Foundation, Prime Ministers Office, Singapore under its National Cybersecurity R&D Program (Grant No. NRF2018NCR-NCR005-0001), the National Satellite of Excellence in Trustworthy Software System (Grant No. NRF2018NCR-NSOE003-0001), the JSPS KAKENHI (Grant Nos. 19K24348, 19H04086), and the Qdaijump Research Program (Grant No. 01277).

ORCID

Fu Song http://orcid.org/0000-0002-0581-2679
Yusi Lei https://orcid.org/0000-0002-3264-8819
Sen Chen https://orcid.org/0000-0001-9477-4100
Lingling Fan https://orcid.org/0000-0002-2428-9297
Yang Liu https://orcid.org/0000-0001-7300-9215

REFERENCES

1. Peng P, Xu C, Quinn L, Hu H, Viswanath B, Wang G. What happens after you leak your password: understanding credential sharing on phishing sites. In: Proceedings of the ACM Asia Conference on Computer and Communications Security, Auckland; July 09–12, 2019:181–192.
2. Mathews L. Phishing scams cost American businesses half a billion dollars a year; 2017. https://www.forbes.com/sites/leemathews/2017/05/05/phishing-scams-cost-american-businesses-half-a-billion-dollars-a-year. Accessed August, 2019.
3. Oest A, Safaei Y, Doupé A, Ahn GJ, Wardman B, Tyers K. PhishFarm: a scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists. In: Proceedings of the IEEE Symposium on Security and Privacy, San Francisco, CA; May 19-23, 2019:1344–1361.
4. Afroz S, Greenstadt R. PhishZoo: detecting phishing websites by looking at them. In: Proceedings of the 5th IEEE International Conference on Semantic Computing, Palo Alto, CA; September 18-21, 2011:368–375.
5. Whittaker C, Ryner B, Nazif M. Large-scale automatic classification of phishing pages. In: Proceedings of the Network and Distributed System Security Symposium, San Diego, CA; February 28–March 3, 2010.
6. Chou N, Ledesma R, Teraguchi Y, Mitchell JC. Client-side defense against web-based identity theft. In: Proceedings of the Network and Distributed System Security Symposium, San Diego, CA; 2004.
7. Zhang Y, Hong JI, Cranor LF. Cantina: a content-based approach to detecting phishing web sites. In: Proceedings of the 16th International Conference on World Wide Web, Banff, Alberta; May 8-12, 2007:639–648.
8. Xiang G, Hong JI, Rosé CP, Cranor LF. CANTINA.: a feature-rich machine learning framework for detecting phishing web sites. *ACM Trans Inf Syst Secur.* 2011;14(2):21:1–21:28.
9. Corona I, Biggio B, Contini M, et al. Deltaphish: detecting phishing webpages in compromised websites. In: Proceedings of the 22nd European Symposium on Research in Computer Security, Oslo; September 11–15, 2017:370–388.
10. Ubing AA, Jasmi SKB, Abdullah A, Jhanjhi N, Supramaniam M. Phishing website detection: an improved accuracy through feature selection and ensemble learning. *Int J Adv Comput Sci Appl.* 2019;10(1):252–257.
11. Microsoft. Microsoft defender smartscreen; 2020. https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-smartscreen/windows-defender-smartscreen-overview. Accessed August, 2019.
12. Srndic N, Laskov P. Practical evasion of a learning-based classifier: a case study. In: Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA; May 18–21, 2014:197–211.
13. Chen S, Xue M, Fan L, et al. Automated poisoning attacks and defenses in malware detection systems: an adversarial machine learning approach. *Comput Secur.* 2018;73:326–344.
14. Duan Y, Zhao Z, Bu L, Song F. Taking care of the discretization problem: a comprehensive study of the discretization problem and a black-box adversarial attack in discrete integer domain. *CoRR*. 2020; abs/1905.07672.
15. Chen G, Chen S, Fan L, et al. Who is real Bob? adversarial attacks on speaker recognition systems. In: Proceedings of the 42nd IEEE Symposium on Security and Privacy; May 23–27, 2021.
16. Zhang L, Wang X, Lu K, Peng S, Wang X. An efficient framework for generating robust adversarial examples. *Int J Intell Syst.* 2020;35(9):1433–1449.
17. Liu WW, Song F, Zhang THR, Wang J. Verifying ReLU neural networks from a model checking perspective. *J Comput Sci Technol.* 2020;35(6):1365–1381.
18. Wan W, Zhang Z, Zhu Y, Zhang M, Song F. Accelerating robustness verification of deep neural networks guided by target labels. *CoRR*. 2020; abs/2007.08520.
19. Xu J, Du Q. Adversarial attacks on text classification models using layer-wise relevance propagation. *Int J Intell Syst.* 2020;35(9):1397–1415.
20. Marchal S, Armano G, Grondahl T, Saari K, Singh N, Asokan N. Off-the-hook: an efficient and usable client-side phishing prevention application. *IEEE Trans Comput.* 2017;66(10):1717–1733.
21. Thomas K, Grier C, Ma J, Paxson V, Song D. Design and evaluation of a real-time url spam filtering service. In: Proceedings of the 32nd IEEE Symposium on Security and Privacy, Berkeley, CA; May 22–25, 2011:447–462.
22. Akhawe D, Felt AP. Alice in Warningland: a large-scale field study of browser security warning effectiveness. In: Proceedings of the 22th USENIX Security Symposium, Washington, DC; August 14–16, 2013: 257–272.
23. Mozilla. Security/safe browsing; 2019. https://wiki.mozilla.org/Security/Safe_Browsing. Accessed August, 2019.

24. Wang Z, Bovik AC. Mean squared error: love it or leave it? a new look at signal fidelity measures. *IEEE Signal Process Mag.* 2009;26(1):98–117.
25. Chai T, Draxler RR. Root mean square error (RMSE) or mean absolute error (MAE)?-arguments against avoiding RMSE in the literature. *Geosci Model Dev.* 2014;7(3):1247–1250.
26. Liu W, Deng X, Huang G, Fu AY. An antiphishing strategy based on visual similarity assessment. *IEEE Internet Comput.* 2006;10(2):58–65.
27. Lei Y, Chen S, Fan L, Song F, Liu Y. Pelican; 2020. https://sites.google.com/view/xpelican. Accessed August, 2020.
28. Krombholz K, Hobel H, Huber M, Weippl E. Advanced social engineering attacks. *J Inf Secur Appl.* 2015;22:113–122.
29. Gupta B, Arachchilage NA, Psannis KE. Defending against phishing attacks: taxonomy of methods, current issues and future directions. *Telecommun Syst.* 2018;67(2):247–267.
30. Chikofsky EJ, Cross JH. Reverse engineering and design recovery: a taxonomy. *IEEE Softw.* 1990;7(1):13–17.
31. Wang X, Yu H. How to break MD5 and other hash functions. In: Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus; May 22–26, 2005:19–35.
32. Xu W, Qi Y, Evans D. Automatically evading classifiers: a case study on PDF malware classifiers. In: Proceedings of the 23rd Annual Network and Distributed System Security Symposium, San Diego, CA; February 21–24, 2016:21–24.
33. Papernot N, McDaniel PD, Goodfellow IJ, Jha S, Celik ZB, Swami A. Practical black-box attacks against machine learning. In: Proceedings of the ACM on Asia Conference on Computer and Communications Security, Abu Dhabi; April 2–6, 2017:506–519.
34. PhishTank. 2019. https://www.phishtank.com. Accessed August, 2019.
35. Phishnet. 2019. http://203.80.17.57/phishnet/individual. Accessed August, 2019.
36. He W, Akhawe D, Jain S, Shi E, Song DX. ShadowCrypt: encrypted web applications for everyone. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ; November 3–7, 2014:1028–1039.
37. Depottools. Depottools; 2019. https://www.chromium.org/developers/how-tos/depottools. Accessed August, 2019.
38. Ninja. 2019. https://github.com/ninja-build/ninja. Accessed August, 2019.
39. Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P. Adversarial examples for malware detection. In: Proceedings of the 22nd European Symposium on Research in Computer Security, Oslo; September 11–15, 2017:62–79.
40. Carlini N, Wagner D. Adversarial examples are not easily detected: bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX; November 3, 2017:3–14.
41. Liang B, Su M, You W, Shi W, Yang G. Cracking classifiers for evasion: a case study on the Google's phishing pages filter. In: Proceedings of the 25th International Conference on World Wide Web, Montreal; April 11–15, 2016:345–356.
42. Saeys Y, Abeel T, Peer dY V. Robust feature selection using ensemble feature selection techniques. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, Antwerp; September 15–19, 2008:313–325.
43. Biggio B, Fumera G, Roli F. Multiple classifier systems for robust classifier design in adversarial environments. *Int J Mach LearnCybern.* 2010;1(1-4):27–41.
44. Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA; May 7–9, 2015.
45. Zhang F, Chan PP, Biggio B, Yeung DS, Roli F. Adversarial feature selection against evasion attacks. *IEEE Trans Cybern.* 2016;46(3):766–777.
46. Yin Z, Wang J, Tang J, Wang W. Defense against adversarial attacks by low-level image transformations. *Int J Intell Syst.* 2020;35(10):1453–1466.
47. Xu W, Evans D, Qi Y. Feature squeezing: detecting adversarial examples in deep neural networks. In: Proceedings of the 25th Annual Network and Distributed System Security Symposium, San Diego, CA; February 18–21, 2018.

48. Sharif M, Bhagavatula S, Bauer L, Reiter MK. Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Vienna; October 24–28, 2016:1528–1540.

How to cite this article: Song F, Lei Y, Chen S, Fan L, Liu Y. Advanced evasion attacks and mitigations on practical ML-based phishing website classifiers. *Int J Intell Syst*. 2021; 36:5210–5240. https://doi.org/10.1002/int.22510