Tear Off Your Disguise: Phishing Website Detection using Visual and Network Identities

Zhaoyu Zhou[1,2], Lingjing Yu[1,2⋆], Qingyun Liu[1], Yang Liu[1], and Bo Luo[3]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] Department of EECS, The University of Kansas, Lawrence, KS, 66045, USA
{zhouzhaoyu, yulingjing, liuqingyun, liuyang}@iie.ac.cn, bluo@ku.edu

Abstract. Adversaries create *phishing websites* that spoof the visual appearances of frequently used legitimate websites in order to trick victims into providing their private information, such as bank accounts and login credentials. Phishing detection is an ongoing combat between the defenders and the attackers, where various defense mechanisms have been proposed, such as blacklists, heuristics, data mining, etc. In this paper, we present a new perspective on the identification of phishing websites. The proposed solution, namely PhishFencing, consists of three main steps: (1) filtering: a list of trusted and non-hosting websites is used to eliminate pages from legitimate hosts; (2) matching: a sub-graph matching mechanism is developed to determine if an unknown webpage contains logo images of whitelisted legitimate websites–once a match is detected, the unknown webpage is considered a *suspicious page*; (3) identification: host features are utilized to identify whether a suspicious webpage is hosted on the same cluster of servers as the corresponding legitimate pages–if not, the suspicious page is tagged as *phishing*. Compared with existing approaches in the literature, PhishFencing introduces an autonomous mechanism to replace the manual process of collecting and refreshing groundtruth data. As a in-network solution, PhishFencing could also partially detect phishing pages hosted on HTTPS servers, without requiring any support from clients. Through intensive experiments, we show that PhishFencing is very effective in comparing with the literature.

Keywords: Phishing · Phishing identification · Website fingerprints.

1 Introduction

Phishing websites forge frequently used, legitimate sites to lure users to submit their sensitive personal data or account credentials. Statistics from the Anti-Phishing Working Group [3] show that most of the phishing websites target at payment (33.0%) or financial (14.3%) sites. Meanwhile, with the increasing popularity of online shopping and banking over the last two decades, their client base has grown from technophiles to normal users, who are less capable of recognizing well-designed phishing sites. Victims deceived by phishing websites often

⋆ Zhaoyu Zhou and Lingjing Yu are co-first authors of this paper.

suffer from serious consequences such as identity thefts and huge property losses. Therefore, from both security research and practice perspectives, it is crucial to efficiently and effectively identify and block phishing websites over the Internet.

Online phishing has been an active research area in the last 10 to 15 years [16, 17], during which both the attack and defense techniques evolve simultaneously. Existing phishing detection methods could be roughly classified into three categories: (1) URL-based (e.g., identifying cloaked URLs), (2) network-based (e.g., detecting DNS poisoning or abnormal DNS registrations), and (3) content-based (e.g., identifying suspicious websites that are visually similar to benign sites). In the battle of online phishing between attackers and defenders, phishing identification methods proposed in the literature may soon become ineffective, for instance, when adversaries purposefully modify page contents or further tampering with phishing URL, such as using "squatting" domains [26].

In this paper, we present a phishing website detection mechanism, named PhishFencing, which attempts to detect discrepancies among a set of relatively robust network and content features. In particular, we identify the "visual identity" of the unknown page, which is often the forged identity, from visual features such as the logos on the page screenshots. We also identify the "network identity" of the unknown page based on its host features, such as IP, AS and geolocation. When both identities are inconsistent, the unknown page is highly likely to be a phishing webpage. The proposed mechanism does not require any support or software installation on the client side. PhishFencing will be deployed at primary exit routers of enterprise networks or at the ISPs, to monitor incoming traffic and to block any phishing pages from flowing into the network.

The proposed approach consists of three main steps, namely *filtering*, *matching*, and *identification*. We first collect the logos of the legitimate websites on the whitelist and generate a fingerprint of visual features for each logo. When a user inside the network visits a webpage, we first invoke the filters to identify if the user is visiting a known trusted website (not necessarily the whitelisted sites). If the visited page comes from a unknown site or web hosting site, we move to the matching step to render the page from passively eavesdropping the data stream. Sub-image matching is invoked to compute the visual similarities between the unknown page and all logos in the local fingerprint database and then compare with a threashold. In this step, the unknown page may trigger matches with fingerprints of multiple legitimate pages, since some legitimate sites may use slightly different logos across several (entry) pages, such as https://www.amazon.com/ and https://www.amazon.co.jp/. In this situation, we pass all the matched legitimate sites as the target websites to the next step. In the identification step, we first extract the network attributes of the hosts of the unknown website and the target websites. After clustering the hosts of the target sites, we finally identify whether the unknown webpage comes from an outlier host, compared to all the clusters of legitimate hosts.

In practice, it is difficult for visual-similarity-based detectors to maintain a complete and up-to-date database/whitelist of all protected legitimate sites, especially consider the fact that the visual layout of the legitimate websites

may change. It is tedious and labor-intensive to ask system administrators to manually monitor all the sites on the whitelist and keep an updated image database. To tackle this challenge, we propose to utilize search engines to collect and update the groundtruth data. With this method, we are able to collect a larger groundtruth set with more comprehensive coverage of the the visual appearances of the whitelist sites.

The main contributions of this paper are three-fold: (1) we propose a novel and highly practical approach to autonomously collect/refresh logo images and visual features from the whitedlisted legitimate websites; (2) we propose the first approach that is able to partially identify phishing websites hosted on HTTPS servers without requiring any interaction with the client computer/browser; and (3) we have developed a three-stage approach, namely PhishFencing, to identify phishing webpages based on the visual and network features that show higher reliability in practice. Through intensive experiments, we demonstrate the superior performance of PhishFencing.

The rest of this paper is organized as follows: we first define the problem and discuss the design goals in Section 2. We introduce the core algorithms and the implementation details of PhishFencing in Sections 3 and 4. We then present the experiment results and performance analysis in Section 5. Finally, we discuss the related works in Section 7 and conclude the paper in Section 8.

2 Problem and Objectives

In this paper, we tackle the problem of discovering and identifying phishing websites, given a whitelist of legitimate websites. Formally, we have a collection of whiltlisted websites as $T = \{T_1, T_2, ..., T_n\}$, in which T_i denotes a *known legitimate website*[4]. In the **threat model**, the adversaries would imitate the visual appearances of a legitimate site T_i, and attempt to trick victims (users) to visit the phishing page and provide their credentials. A user from within the enterprise network visits an external page S_i (i.e., the *unknown page*), which could be a phishing page that might bring potential damage to the enterprise network. The objective of this project is to design a phishing detection mechanism $M(S, T)$ that, giving a new website S_x, identifies whether it is a phishing website imitating T_x: $M(S_x, T_x) = \{0, 1\}$.

In this project, we aim to tackle two practical challenges: (1) Groundtruth data collection and refresh: the whitelist of legitimate websites usually contains a list of site names (e.g., Bank of America) and/or their entry URLs (e.g., https://www.bankofamerica.com/). Moreover, each legitimate website may have multiple entry points besides the root page, e.g., BoA have pages like https://www.bankofamerica.com/credit-cards/manage-your-credit-card-account/. It is practically impossible to manually visit all these sites to generate visual fingerprints, and to keep all the fingerprints up to date. In PhishFencing, we employ web search engines to crawl a set of logos of legitimate pages for each whitelist

[4] In this paper, we use whitelisted sites and legitimate sites interchangeably.

entry, with the hypothesis that the top results from the largest commercial search engine are *trustworthy*. (2) Encrypted traffic: to the best of our knowledge, all existing phishing detection mechanisms for HTTPS phishing sites require collaboration from the client side, such as installation of browser add-ons, or local detection mechanisms. However, it is impractical to require and enforce that all the devices connected to the network to have anti-phishing software/client installed. Especially, with the growing popularity of BYOD (bring your own device) programs in the industry, more personal devices are connected to corporate networks. In PhishFencing, we present the first mechanism to (partially) detect phishing pages hosted on HTTPS sites without requiring any assistance from the client computer/browser.

3 Features and Algorithms

In this section, we first introduce features utilized in PhishFencing, and then describe the core algorithms for image matching and phishing detection.

3.1 Features

We aim to extract features which are easily obtained and difficult to manipulate by attackers. For example, URLs and content of webpages (HTML codes and resources) are not stable enough and easy to be bypassed by attackers. For URLs, adversaries may use squatting domain [26] to imitate target sites' URLs, while others construct normal but totally irrelevant URLs to overpass detection [2]. In the case of content of webpages, some adversaries use exactly the same HTML structures and resources as the target websites, while others carefully manipulate those content to overpass detection. At the same time, most of legitimate websites, such as Amazon, change texts and pictures on their webpages frequently, which also makes content features less reliable. Features used in PhishFencing are listed in Table 1. Next, we will describe each feature in detail.

Table 1: Features used for PhishFencing.

#	Feature	Step	#	Feature	Step
1	domain	filtering	5	IP prefix	identification
2	form	filtering	6	AS number	identification
3	logo	matching	7	geolocation	identification
4	webpage screenshot	matching			

Domain features. PhishFencing takes the host names of HTTP pages or Server Name Indication extensions (SNI) of HTTPS sites as the domain feature. We assume that pages from Alexa top sites are benign. In practice, we crawl the domain names of Alexa top 3000 sites, and denote them as the list of *trusted websites* (not to be confused with the whitelist of legitimate websites). Note that

pages from web hosting service providers, such as https://sites.google.com, are not all trustworthy, since they have been found to be utilized to host phishing webpages in the literature [26]. Therefore, we exclude all web hosting services from the trusted site list. Except for the web hosting services, we can safely assume that adversaries are unable to allocate sub-domains of the highly popular, heavily monitored, and better managed sites to host phishing pages. Compromised domains, as exceptions to this assumption, are discussed in Section 6.

Form features. Forms, including INPUT and FORM tags, are used to collect information from the client side. When an HTML file does not contain any form element, it cannot be used to harvest personal information [28].

Logo features. Logos are used in phishing detection in the literature, such as PhishZoo [4]. However, it is tedious and labor-intensive to manually discover and refresh all logo images of whitelisted sites. To overcome this drawback, PhishFencing automatically collects and updates logo images using search engines. In practice, a query consists of the websites' name plus the keyword "logo", e.g., "paypal.com logo", is sent to the search engine. The top n results from picture search are crawled to enhance the diversity of the result set, since a site may have multiple versions of logos and they may be presented differently in images.

Webpage screenshot features. Different from PhishZoo [4], PhishFencing uses the screenshot of an unkown webpage in matching with logos from whitelisted sites, rather than exhaustively comparing with every image on the webpage, for two reasons: (1) repetitively invoking the matching algorithm to compare every image from the unknown page against the fingerprint of every logo is computationally expensive; and more importantly (2) adversaries may use tricks to avoid using full/original logo images to avoid detection, e.g., splitting the logo into small images, or overlay layers of images. However, they still need to preserve the overall visual presentation of the spoofed page. Hence, we render the full pages and utilize sub-image matching to compare them with logo fingerprints. In practice, we use Selenium to capture a 1920×1080 screenshot for each unknown page. Note that the identities (logos) of spoofed sites are always presented at the top of the page, hence, it is not necessary to capture the entire page.

Host features. We treat all IP prefix features, Autonomous System number (AS number) features, and geolocation features as host features. Host features are also widely adopted in phishing webpage detection. Host features are considered as relatively reliable. It is difficult for attackers to compromise servers hosting legitimate websites, hence, the host distribution of phishing websites should be different from that of legitimate websites. Note that PhishFencing passively collects IP addresses of unknown and legitimate sites from the same channel, i.e. an ISP or a gateway of enterprise network. This ensures the consistency of observed IP distribution. For a given IP address, PhishFencing collects its AS number and the server's geographic location using the *MAXMIND* database [1]. The IP prefix is extracted to represent the class C network the IP belongs to, for the reason that prefixes contain IP addresses association information [27]. For a whitelisted legitimate website, we collect IP address prefixes, AS numbers, latitudes and longitudes features, to be used to train a model for outlier detection.

3.2 Algorithms

In this paper, we employ a graph matching algorithm to decide whether a logo image is a sub-graph of a screenshot, as well as an classification algorithm to identify phishing websites.

Graph Matching Algorithm. As the phishing webpage can be self-defined, attackers can use different scales of logo images to deceive users and to evade logo detection methods which are not robust enough on image scale variation. So we applied Scale Invariant Feature Transform (SIFT) algorithm [19] which can generate scale-invariant keypoint descriptors.

The major steps are briefly explained as follows. The first step is to detect extrema in the scale-space. To achive this goal, *SIFT* generates smoothed images in different scale, defined as $L(x, y, \sigma)$. Given a 2D image $I(x, y)$, $L(x, y, \sigma)$ is computed from the convolution of a variable-scale Gaussian $G(x, y, \sigma)$ and $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{1}$$

where $*$ refers to the convolution operation, x and y are the spatial coordinates of a plane and:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2}$$

Then through difference-of-Gaussian function, scale-space extrema, which is regarded as potential interest points, can be detected. The second step focus on locating keypoints accurately. From potential interest points extracted in the first step, *SIFT* rejects the points which have low contrast and are poorly localized along an edge for stability. Next, based on local image gradient directions, *SIFT* assigns one or more orientations to each keypoint location. In the last step, *SIFT* set a region around each keypoint's location where some points sampled and the gradient magnitude and orientation of these points are computed to form a $4 * 4 * 8$ vector as the keypoint descriptor.

In particular, when using different background colors, such as white or black, attackers need to invert the color of logo images accordingly for users to recognize. When color inverted, the keypoints' positions could still match but the keypoints orientation and descriptor vector would change, which will reduce our matching performance. So we use both original image and color inverted image for matching. To be specific, we firstly convert a BGR image to a GRAY image, then for all x and y in the image plane, we compute $I'(x, y) = 255 - I(x, y)$. So for a logo image, we generate two sets of keypoint descriptors for matching. We will compare the performance of using color inverted images or not in Secction 5.

After keypoint descriptors generated, we applied Fast Library for Approximate Nearest Neighbors (FLANN) algorithm [24] which will build up index trees (multiple randomized kd-trees in practice) for screenshots' keypoints to find the nearest neighbor in a screenshot for each keypoint in logo image. The nearest neighbor refers to the keypoint with minimum Euclidean distance from the keypoint descriptor vector. For each keypoint in logo image P_{logo}, the index tree is used to locate it's nearest keypoint in the screenshot $P_{screenshot}$. In order to

evaluate the matchness between P_{logo} and the corresponding $P_{screenshot}$, we utilize the secondary neighbor keypoint $P'_{screenshot}$, to calculate the ratio $R_{matching}$ of distances:

$$R_{matching} = \frac{D(P_{logo}, P_{screenshot})}{D(P_{logo}, P'_{screenshot})} \tag{3}$$

where D refers to the Euclidean distance.

As the correct matches need to make the nearest neighbor significantly closer than the secondary neighbor which refers to the closest incorrect match, we can reject matches with low distance ratio R [19]. Then we calculate the percentage of keypoints in logo images, which own correct matches Sim, to decide whether the logo image is the sub-graph of the screenshot. Higher Sim means that more keypoints are correctly matched and so logo images have higher possibility as the sub-graph of the screenshot. So when the Sim is higher than a threshold, we say the match between a logo image and a screenshot is achieved.

Phishing Website Identification Algorithm. To identify phishing websites, we apply host features of both target websites and suspecious websites to One Class Support Vector Machine (one-class SVM) [9] to detect outliers, which are host features of phishing websites. One-class SVM uses only positive data, i.e. host features from the target website, as input to estimate the support vector of a high-dimensional distribution. Given a target website, our training vectors can be constructed as $\mathbf{h}_i = f_{1,i}, \ldots, f_{3,i}$ where $i = 1, \ldots, m$, m is the number of webpages, $f_{t,i}$ denotes the tth feature in host features and $\mathbf{h}_i \in R^n$ presents the host features extracted from the ith webpage. During the training process, we need to find out ω and $\mathbf{b}$ satisfied:

$$\begin{aligned} \min_{\omega,\mathbf{b}} \frac{1}{2}||\omega||^2 + \frac{1}{\nu m}\sum_{i=1}^{m}\epsilon_i \\ s.t.\ \omega^T\mathbf{h}_i + \mathbf{b} \geq 1 - \epsilon_i \\ \epsilon_i \geq 0, i = 1, 2, \ldots, m \end{aligned} \tag{4}$$

where ω and $\mathbf{b}$ are used to construct the *hyperplane* which is the boundary of positive data. Since our training data can not be linearly separated, kernel function Radial Basis Function kernel (RBF kernel) is employed to map the data to a higher dimension feature space, in which data can be linearly separated. For two samples $\mathbf{h}, \mathbf{h}'$, the RBF kernel $K(\mathbf{h}, \mathbf{h}')$ can be defined as:

$$K(\mathbf{h}, \mathbf{h}') = exp(\frac{||\mathbf{h} - \mathbf{h}'||_2^2}{-2\sigma^2}) \tag{5}$$

Once ω and $\mathbf{b}$ are optimized, given a new vector $\mathbf{h} = f_1, \ldots, f_3$, if $\mathbf{h}$ satisfied:

$$\omega^T\mathbf{h} + \mathbf{b} < 0 \tag{6}$$

then we regard this new vector as an outlier, i.e. host features from a phishing website.

4 Design of PhishFencing

4.1 Overview of PhishFencing

As shown in Figure 1, PhishFencing consists of three steps: (1) in the filtering step, we apply domain features (Alexa top 3000 domains) and form check to filter out trusted and harmless websites. The remaining pages are called the *unknown webpages*. (2) In the matching step, PhishFencing checks whether the whitelisted legitimate websites' logo images are sub-graphs of screenshots of unknown webpages. If a logo image is identified as a sub-graph of the screenshot of a webpage, we regard the webpage as a *suspicious webpage*. (3) Finally, PhishFencing applies outlier detection on host features of these *suspicious webpages* to identify phishing webpages.

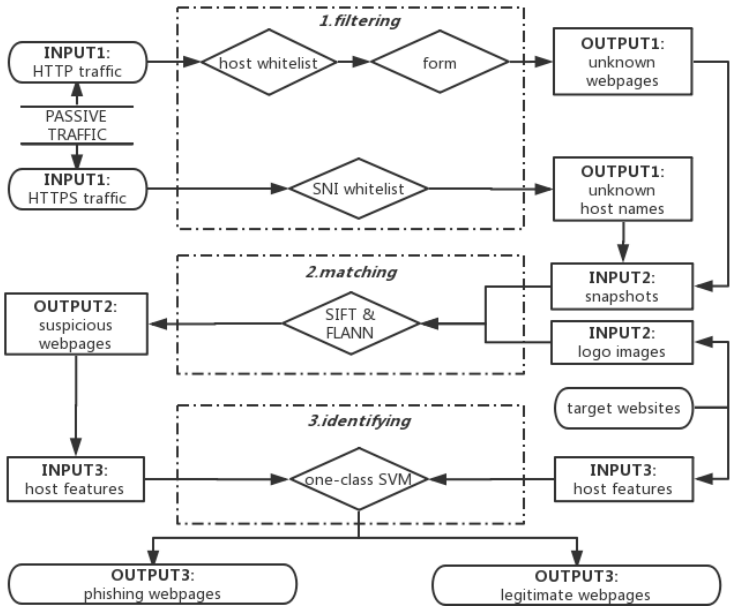


Fig. 1: Model of PhishFencing

4.2 Filtering

In filtering step, PhishFencing attempts to employ simple heuristics to eliminate websites that are definitely not phishing websites. First, PhishFencing collects HTTP/HTTPS streams passively. From HTTP packets, host names, HTML files and URLs can be extracted. Hence, domain features and form check can be applied directly to filter out trusted websites in two heuristics: (1) Suffix

matching is applied on domain features to identify if the unknown page comes from a trusted website, as introduced in Section 3.1. (2) The HTML page and all sub-frames are scanned to identify forms. When a webpage does not contain any form, it cannot be a phishing page. In practice, these two heuristics eliminate majority of the unknown pages with very small computation cost. Note that this step is only introduced to save computation. In an environment where computing resource is not a concern, we can reduce size of the trusted sites list, just in case an adversary compromises an Alexa top domain (or its sub-domain) to host phishing pages. On the other hand, for HTTPS streams, we can neither access the complete URLs nor the HTML source files since they are all encrypted. Hence, PhishFencing only applies domain-based filtering on the SNI field, which indicates the host name of a website, to eliminate trusted websites from going into future steps.

4.3 Matching

In the matching step, PhishFencing identifies suspicious webpages based on the similarities between the rendered unknown pages and logo images of whitelisted websites. HTTP and HTTPS pages are handled differently in this step.

For HTTP pages, PhishFencing renders the *unknown webpage* fetched from passive HTTP streams, and then captures a screenshot image of the fully rendered page. Simultaneously, PhishFencing fetches logo images of whitelisted websites as described in Section 3.1, and pass them to SIFT matching.

For HTTPS pages, PhishiFencing could only extract host names from the packets, not the complete URL or any file name/content, hence, PhishFencing cannot directly obtain the corresponding webpages. To (partially) solve this problem, for each host name from the HTTPS streams, PhishFencing searches it on search engines and crawls all the returned URLs within the domain. For example, we search "bit.ly" which is extracted from the SNI field of the HTTPS packet, and we can see URLs such as "https://bit.ly/2kIChZC" shown up in results. All the returned URLs are actively crawled to obtain the screenshots of the corresponding webpages[5]. At the same time, PhishFencing also visits the host name directly to crawl the default (root) page of the domain, and follows any link on the page to collect all accessible pages in the domain. All pages are rendered and screenshots are captured. The rationale of these operations is that the attackers often post phishing URLs on other websites (such as online forums) so that they can reach out to a larger audience of potential victims. Such URLs are likely to be captured by web search engines. Meanwhile, we also see that many (sub)domains are only created for phishing purposes–once a confirmed phishing page is found from the domain, especially as the root page of the domain or accessible from the root page, other pages in the same domain become highly suspicious.

[5] An upper limit of crawled URLs is set just in case the domain is huge, however, it is rarely reached in our experiments.

Next, the SIFT algorithm is invoked to generate keypoint descriptors of both logo images and screenshots. The keypoint descriptors of each logo image and screenshot pair are further sent to the FLANN algorithm to determine whether the logo image matches a sub-graph of the screenshot image. When a screenshot contains sub-graphs that are similar to a logo image in the whitelist, the corresponding page is then marked as *suspicious*, which is sent to the next step for further identification.

4.4 Identification

In the previous step, PhishFencing has discovered suspicious webpages, whose visual identities carry significant similarity with whitelisted sites. In the identification step, PhishFencing attempts to finally determine whether a suspicious page is a phishing page based on the host features, i.e., by comparing the host distribution of the suspicious page and the whitelisted legitimate pages.

As described in Section 3.1, PhishFencing collects the IP addresses of the websites which host suspicious webpages, and the IP addresses of the corresponding legitimate websites. We then employ *MAXMIND* to obtain the AS numbers and geolocation of these IP addresses. Finally, we utilize `one-class SVM` on the host features of both the suspicious websites and their corresponding legitimate websites to discover outliers. All outliers are then labeled as phishing webpages, which should be blocked at the firewalls.

5 Experimental Evaluation

In this section, we empirically evaluate PhishFencing and demonstrate its performance. We first describe our dataset. Then we define the evaluation metrics and present the experiment results.

5.1 Dataset

Logo Fetching Mechanism. we chose domains of Alexa top 1600 sites to evaluate the effectiveness of our logo fetching mechanism. We deployed *Google Images Download* to obtain the first 10 images for each domain from Google. At the same time we took screenshot of the root page (the landing page when directly visit a domain) for each domain by *Selenium*. Since some websites apply bot detection technologies such as reCaptcha to avoid crawlers, we verified the correctness of logos manually.

PhishFencing. We use *PhishTank* as the source of phishing pages. URLs from *PhishTank* are manually verified to exclude links that land on irrelevant webpages or with 404 errors. To obtain the groundtruth dataset, we visited verified *PhishTank* URLs from computers inside our institutional network. We captured the traffic using *tcpdump* at the gateway, and used them as positive (phishing)

samples. Similarly, we visited the corresponding legitimate websites to generate negative (non-phishing) samples. For each legitimate site, we intended to visit multiple webpages in different content, HTML structures, languages, and background colors to increase the diversity of the negative samples, and to accumulate IP address features of the legitimate sites.

The groundtruth dataset has been collected for 7 days continuously with 77,539 phishing URLs verified by *PhishTank*, among which 13,902 were labelled with target brands. We followed SquatPhish [26] to select 8 most frequently targeted brands, which cover 68.98% of the phishing webpages in our groundtruth dataset. They are *paypal*, *microsoft*, *facebook*, *google*, *amazon*, *apple*, *dropbox*, and *yahoo*. Since PhishFencing uses an autonomous mechanism to collect groundtruth data, it could easily scale up to handle thousands of whitelisted sites. After manually verified these phishing webpages based on the method mentioned in [26], only 772 URLs remained as valid phishing URLs (majority of the phishing websites went offline after a very short lifespan), in which 48.7% are hosted on HTTP and 51.3% are hosted on HTTPS.

For each brand, we chose its primary website(s) from Alexa as our target website(s). For brands like Amazon, multiple target site have been identified, such as amazon.com, amazon.cn, amazon.jp, etc. Note that in our paper, if two host names have the same second-level domain (SLD) and the same top-level domain (TLD), they are considered to belong to the same site. For example, "*scholar.google.com*" and "*www.google.com*" belong to the same website "*google.com*" according to our definition. For each target website, PhishFencing crawled the top 10 logo images using *Selenium* with *chromedriver*, and eliminated duplicate logos (logos with similar SIFT features), to generate the set of logo images. Meanwhile, for all the target websites, 461 different IP addresses were extracted by PhishFencing to build host features.

5.2 Evaluation Metrics

The overall performance is measured in terms of precision ($P_{overall}$) and recall ($R_{overall}$) where

$$P_{overall} = \frac{|\{phising\ webpages\} \cap \{identified\ webpages\}|}{|\{identified\ webpages\}|}, \tag{7}$$

$$R_{overall} = \frac{|\{phising\ webpages\} \cap \{identified\ webpages\}|}{|\{phishing\ webpages\}|}. \tag{8}$$

We also employed the F1-score to combine both precision and recall to evaluate the overall effectiveness of different approaches. The F1-score is defined as:

$$F_1 = \frac{2 \times P_{overall} \times R_{overall}}{P_{overall} + R_{overall}} \tag{9}$$

At the same time, since PhishFencing consists of three primary steps, we also want to evaluate each steps separately to see their best performance. In

the filtering step, we can simply adjust the list of trusted websites to ensure all potential phishing webpages are passed to the following steps. In the matching step, we first evaluate the reliability of our automatic logo fetching mechanism, we define the accurate rate (A_{logo}) on logo retrieving as:

$$A_{logo} = \frac{|\{websites\ with\ logo\ correctly\ fetched\}|}{|\{websites\}|}. \tag{10}$$

As for PhishFencing's matching performance, we define *matching precision* ($P_{matching}$) and *matching recall* ($R_{matching}$) to describe the performance of subgraph matching.

$$P_{matching} = \frac{|\{webpages\ with\ certain\ logo\} \cap \{matched\ webpages\}|}{|\{matched\ webpages\}|}, \tag{11}$$

$$R_{matching} = \frac{|\{webpages\ with\ certain\ logo\} \cap \{matched\ webpages\}|}{|\{webpages\ with\ certain\ logo\}|}. \tag{12}$$

Last, we evaluate the performance of the identification step with samples that are correctly matched. We define identification precision as $P_{identify}$ and identification recall as $R_{identify}$ in a very similar way as Equations 7 and 8.

5.3 Performance Evaluation

In this section, we first present the reliability of our logo fetching mechanism. Then we evaluate PhishFencing's performance on groundtruth dataset.

Effectiveness of Logo Retrieval. We evaluated the performance of our automatic logo fetching mechanism through manual verification: (1) we utilized the logo fetching mechanism to retrieve the logo images of Alexa's top 1600 websites; (2) we also downloaded the screenshots of each domain's landing page; (3) for each of the top 1600 sites, we manually verified if the fetched logo appears in the landing page. For domains which we were unable to retrieve the right logo images, we further examine the errors and categorized them, as shown in Figure 3.

As shown in Figure 3, for 4.43% of the websites, the fetched logos do not appear on the domains' landing pages, while the landing pages appear to be legitimate (Error type #1). Meanwhile, we were unable to download legitimate landing pages for some domains: (Error type #2) the landing pages are not reachable due to DNS error, 404 page not found error, or connection time-out. (Error type #3) Landing pages of some domains behave maliciously such as browser hijacking. (Error type #4) Some domains instantly redirect the browser to other domains, hence, the original domains do not host any service. (Error type #5) Some domains were shut down while sale or notification pages were reached. (Error type #6) There are also domains used for ad serving, which work as connectors between website owners and advertisers. And (Error type #7) some

#	Error type	Error rate
1	Logo mismatch	4.43%
2	Domain unreachable	3.56%
3	Browser hijacker	2.75%
4	Redirect	2.75%
5	Shut down	1.06%
6	Ad serving	0.44%
7	No logo	0.25%

Fig. 3: Causes and frequency of failed/wrong logo image retrieval.

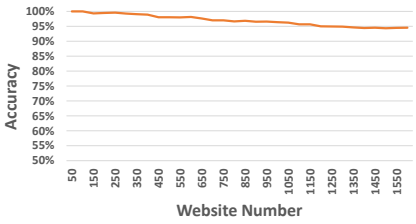


Fig. 4: Accuracy of logo fetching on Alexa's top k websites.

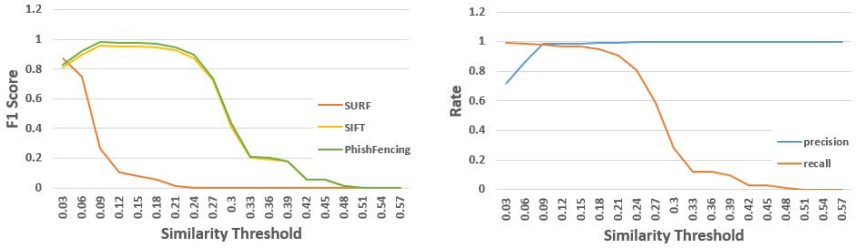
websites do not have any logo. Domains that generate errors #2 to #6 do not provide web services, hence, we eliminated them in our further evaluations.

After we eliminated the domains mentioned above (error types #2 to #6), we calculated the accuracy of our logo fetching mechanism for Alexa's top k sites. As shown in Figure 4, PhishFencing correctly fetched the logo images of at least 95% of the top 1600 sites. PhishFencing performs better on websites that rank higher, for example, logo fetching accuracy reaches 98% for top 650 sites.

PhishFencing Evaluation. To present the performance of PhishFencing in matching step, we compared our mechanism with *SIFT* and *SURF* which were employed in *PhishZoo* [4] as shown in Figure 4(a). F1-scores with *SIFT* were much higher than those with *SURF*. And with our improved algorithm, we can slightly outperform the recall and precision of original SIFT. To be more specific, we calculated both precision ($P_{matching}$) and recall ($R_{matching}$) rate of PhishFencing as shown in Figure 4(b). We can see that when $Sim = 0.09$, we can obtain 99.27% precision and 97.90% recall in the matching step. Note that we used names of websites to fetch logo images which is more reliable than using brand names. For example, logo images of "*amazon.cn*" and "*amazon.com* are different. If we simply use "*amazon logo*" to fetch logo images, the logo of "*amazon.cn*" would not shown up in the top results.

In the identification step, PhishFencing achieved 97.8% ($P_{identify}$) precision and 100% recall ($R_{identify}$) on 8 target brands on average using host features from webpages which had been successfully matched. Note that legitimate IP addresses were collected in nearly 2-3 hours for each target website. In Figure 6, we list the number of IP addresses collected on each of the 8 target brands. The number of IP addresses are not necessarily massive which suggest that our mechanism is not depending on large amount of prior data and can be used on client side as well.

As for the overall performance, we compared PhishFencing with the SquatPhish approach [26], which is the state of art solution for identifying phishing webpages with specific target brand. We applied SquatPhish which is open sourced on github on our groundtruth dataset. As shown in Figure 7, we first



(a) (b)

Fig. 4: Matching performance: (a) F1 -score comparison of SIFT, SURF and the matching mechanism in PhishFencing approach on groudtruth data. (b) Precision and Recall rate with different similarity threshold selected using PhishFencing on groundtruth data.

target website	# of IP addresses
amazon.com	138
apple.com	98
microsoft.com	67
google.com	51
yahoo.com	50
dropbox.com	42
paypal.com	8
facebook.com	7

Fig. 6: Number of legitimate IP addresses collected for each target website in groundtruth data

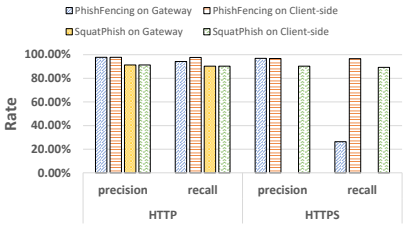


Fig. 7: Overall performance comparison of SquatPhish on HTTP, PhishFencing on HTTP and PhishFencing on HTTPS tested on groundtruth data.

compare the performance of both approaches installed on the gateway to capture HTTP streams. PhishFencing reaches 97.8% precision and 97.7% recall which are both higher than SquatPhish. Then we evaluate the performance of PhishFencing on HTTPS streams. Since SquatPhish utilizes webpage's screenshot and HTML source code which cannot be obtained from encrypted packets, SquatPhish cannot handle HTTPS streams when it is deployed at the gateway. The results show that PhishFencing achieved 26.32% recall on HTTPS-hosted phishing websites, when it is deployed at the gateway and only relies on two side channels to infer if the host domain is suspicious. Although there is still room to improve the recall, PhishFencing is the first solution of its kind to partially detect HTTPS-hosted phishing at the gateway.

Last, we also like to note that PhishFencing performs well on small size of host features. Therefore, it can be deployed on the client side, which only has limited data for the host features of the legitimate sites. In the experiments, PhishFencing's performance on HTTP streams remains high when it is deployed at

the client side. Meanwhile, the recall rate on HTTPS-hosted phishing increased dramatically since we are now able to obtain the full URLs from HTTPS packets.

6 Discussions

PhishFencing is effective in phishing detection in the experiments, however, we still recognize its limitations and opportunities for future improvements.

First, as we have explained in Section 4.3, PhishFencing could only obtain the domain name, not the full URL, from HTTPS streams. Therefore, we rely on two side channels to find (other) pages hosted in this domain. This method appears to be effective on a portion of the HTTPS-hosted phishing websites. However, when no phishing pages are detected through the two side channels, PhishFencing is unable to discover any "hidden" phishing page. While the problem of detecting HTTPS-hosted phishing without the support from the client side is very challenging, we believe there is still space to improve.

We have applied suffix matching on domains of websites to filter out trusted websites, with the assumption that pages hosted on trusted sites (excluding any web-hosting service providers) are trustworthy. However, this assumption may be violated, especially when the adversary compromises a trusted site to host phishing webpages. In response, PhishFencing could cache the visits to trusted domains, and use the non-utilized server cycles to evaluate the (sampled) cached pages. When phishing is identified, *ex post facto* repairing mechanism is invoked, while the corresponding site would be removed from the trusted site list.

For HTTPS-hosted unknown pages, PhishFencing relies on correct host names in SNI fields. However, *domain fronting*, a versatile censorship circumvention technique, can be employed to show one domain in SNI field while using another domain in the HTTP host field [14]. In this way, attackers can replace the host name in the SNI field with a legitimate host name to evade our detection.

Some logo images may be shown on irrelevant websites. For example, the *Visa* logo may be shown on retailers' homepages to show that visa cards are accepted, or on a check-out/payment pages. In the first case, the pages are highly likely to be eliminated from phish detection since they usually do not contain any form. In the second case, a legitimate HTTPS-hosted payment page is unlikely to be misclassified, since the page itself is not accessible to PhishFencing, while the domain is likely to be benign. However, HTTP-hosted pages carrying Visa logos and containing forms (e.g., a retailer's homepage with a input box for search) may be misclassified as phishing. Fortunately, such cases are very rare in our experiments and they can be fixed by adding those sites to the trusted list.

Last, PhishFencing evaluates the visual identities of webpages by comparing the logos of whitelisted sites and the phishing webpages. In the very rare case where a whitelisted site do not have a logo or the logo image is not shown on the phishing webpages, PhishFencing's recall would be impacted. However, in our groundtruth dataset, all the sites in the whitelist have logo images and there are only 2.19% known phishing webpages that do not have any logo on them.

7 Related Works

Phishing website detection mechanisms can be roughly categorized into target-independent and target-dependent approaches. Target-independent approaches extract common features from all the phishing websites to train a model for phishing websites identification [21, 23]. Target-dependent approaches, which PhishFencing belongs to, identify phishing websites mainly through comparing the similarity between target websites and on-identifying websites [28, 31].

For target-independent approaches, the most commonly used features are URL features (i.e. structures and lengths of an URL) [5, 15, 18], webpage features (i.e. links, keywords, and HTML DOM extracted from a webpage) [20] and host features (i.e. IP addresses, AS numbers and geolocation of a website's hosts). Mechanisms in [8,11,22,25,27,30] combine large amount of features mentioned above and employ different machine learning algorithms to detect phishing websites. Apart from these machine learning methods, [10, 13, 29] make use of websites' identities as well as search engines. They try to figure out identities of a website at first. For example, [29] uses Term Frequency Inverse Document Frequency (TF-IDF) to extract terms with highest weight as a website's identities. [13] applies Optical Character Recognition (OCR) on a webpage's screenshot and regards the text generated by OCR as the webpage's identity. [10] uploads segmented screenshot of a webpage to Google Image Search engine and regards the keywords returned as the webpage's identities. Then they query the identities of a website through search engine. If the domain name of the on-identifying website does not match any of N top search result, they would classify the website as a phishing one. However, target-independent approaches use generic characteristics which can be constructed by attackers to evade the detection systems.

For target-dependent methods, visual features such as screenshot and logo image are most commonly used. Meerkat [6] trains deep learning models to detect phishing webpages hosted on compromised websites via visual elements in webpages. Apart from visual elements on the webpages, [26] applies OCR on URLs to detect squatting phishing domains. Besides visual features, [7] compares the layout and HTML text between target webpage and on-identifying webpage. [4, 12] combines HTML features and visual features for identifying.

PhishFencing is different from existing approaches that: (1) PhishFencing chooses visual and network features which are representative and difficult to be manipulated compared to the target-independent methods. (2) PhishFencing utilizes search engines to autonomously collect/refresh logo images and visual features of HTTPS websites. Existing target-dependent approaches either identify logo manually or segmented the screenshot to locate logo which is less reliable than the approach in our mechanism. (3) PhishFetching can deal with phishing websites hosted on HTTPS which, to the best of our knowledge, has not been mentioned by other works.

8 Conclusion

In this paper, we present a phishing website identification approach named PhishFencing. The core idea is to detect if an unknown webpage carries the visual identity (logo) of a whitelisted legitimate site, while its host features deviate from the distribution of the known hosts of the legitimate site. PhishFencing consists three major steps: filtering, matching, and identification. As a network-based solution, PhishFencing will be deployed at the gateways of enterprise networks or at the ISPs' network backbones, to block phishing pages from being transmitted to end users. In the experiments, we demonstrate that PhishFencing outperforms state-of-art phishing detection solutions in the literature.

Acknowledgements

Zhaoyu Zhou, Lingjing Yu, Qingyun Liu, and Yang Liu were supported in part by Y8YY041101 and Y9W0013401. The authors also like to thank the anonymous reviewers for their constructive suggestions.

References

1. Maxmind. https://www.maxmind.com/en/geoip2-databases
2. Phishtank. https://www.phishtank.com/index.php
3. Phishing activity trends report. Tech. Rep. 2nd Quarter, APWG (2018)
4. Afroz, S., Greenstadt, R.: Phishzoo: Detecting phishing websites by looking at them. In: IEEE ICSC. pp. 368–375 (2011)
5. Blum, A., Wardman, B., Solorio, T., Warner, G.: Lexical feature based phishing url detection using online learning. In: ACM AISec Workshop. pp. 54–60 (2010)
6. Borgolte, K., Kruegel, C., Vigna, G.: Meerkat: Detecting website defacements through image-based object recognition. In: USENIX Security. pp. 595–610 (2015)
7. Britt, J., Wardman, B., Sprague, A., Warner, G.: Clustering potential phishing websites using deepmd5. In: USENIX LEET (2012)
8. Canali, D., Cova, M., Vigna, G., Kruegel, C.: Prophiler: a fast filter for the large-scale detection of malicious web pages. In: WWW Conference. pp. 197–206 (2011)
9. Chang, C.C., Lin, C.J.: Libsvm: A library for support vector machines. ACM TIST **2**(3), 27 (2011)
10. Chang, E.H., Chiew, K.L., Tiong, W.K., et al.: Phishing detection via identification of website identity. In: IEEE ICITCS. pp. 1–4 (2013)
11. Choi, H., Zhu, B.B., Lee, H.: Detecting malicious web links and identifying their attack types. WebApps **11**(11), 218 (2011)
12. Corona, I., Biggio, B., Contini, M., Piras, L., Corda, R., Mereu, M., Mureddu, G., Ariu, D., Roli, F.: Deltaphish: Detecting phishing webpages in compromised websites. In: ESORICS. pp. 370–388 (2017)
13. Dunlop, M., Groat, S., Shelly, D.: Goldphish: Using images for content-based phishing analysis. In: IEEE ICIMP. pp. 123–128 (2010)
14. Fifield, D., Lan, C., Hynes, R., Wegmann, P., Paxson, V.: Blocking-resistant communication through domain fronting. PETS **2015**(2), 46–64 (2015)

15. Garera, S., Provos, N., Chew, M., Rubin, A.D.: A framework for detection and measurement of phishing attacks. In: ACM workshop on Recurring malcode (2007)
16. Jagatic, T.N., Johnson, N.A., Jakobsson, M., Menczer, F.: Social phishing. Communications of the ACM **50**(10), 94–100 (2007)
17. Khonji, M., Iraqi, Y., Jones, A.: Phishing detection: a literature survey. IEEE Communications Surveys & Tutorials **15**(4), 2091–2121 (2013)
18. Le, A., Markopoulou, A., Faloutsos, M.: Phishdef: Url names say it all. In: INFOCOM. pp. 191–195 (2011)
19. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
20. Ludl, C., McAllister, S., Kirda, E., Kruegel, C.: On the effectiveness of techniques to detect phishing sites. In: DIMVA. pp. 20–39 (2007)
21. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious urls. In: ACM KDD. pp. 1245–1254 (2009)
22. Marchal, S., Armano, G., Gröndahl, T., Saari, K., Singh, N., Asokan, N.: Off-the-hook: An efficient and usable client-side phishing prevention application. IEEE Trans. on Computers **66**(10), 1717–1733 (2017)
23. Marchal, S., François, J., State, R., Engel, T.: Phishstorm: Detecting phishing with streaming analytics. IEEE Transactions on Network and Service Management **11**(4), 458–471 (2014)
24. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1) **2**(331-340), 2 (2009)
25. Thomas, K., Grier, C., Ma, J., Paxson, V., Song, D.: Design and evaluation of a real-time url spam filtering service. In: IEEE S&P. pp. 447–462 (2011)
26. Tian, K., Jan, S.T., Hu, H., Yao, D., Wang, G.: Needle in a haystack: tracking down elite phishing domains in the wild. In: ACM IMC. pp. 429–442 (2018)
27. Whittaker, C., Ryner, B., Nazif, M.: Large-scale automatic classification of phishing pages (2010)
28. Xiang, G., Hong, J., Rose, C.P., Cranor, L.: Cantina+: A feature-rich machine learning framework for detecting phishing web sites. ACM TISSEC **14**(2) (2011)
29. Xiang, G., Hong, J.I.: A hybrid phish detection approach by identity discovery and keywords retrieval. In: WWW Conference. pp. 571–580 (2009)
30. Zhang, W., Jiang, Q., Chen, L., Li, C.: Two-stage elm for phishing web pages detection using hybrid features. World Wide Web **20**(4), 797–813 (2017)
31. Zhang, Y., Hong, J.I., Cranor, L.F.: Cantina: a content-based approach to detecting phishing web sites. In: WWW Conference. pp. 639–648 (2007)