VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity

Sahar Abdelnabi
CISPA Helmholtz Center for Information Security

Katharina Krombholz
CISPA Helmholtz Center for Information Security

Mario Fritz
CISPA Helmholtz Center for Information Security

ABSTRACT

Phishing websites are still a major threat in today's Internet ecosystem. Despite numerous previous efforts, similarity-based detection methods do not offer sufficient protection for the trusted websites, in particular against unseen phishing pages. This paper contributes *VisualPhishNet*, a new similarity-based phishing detection framework, based on a triplet Convolutional Neural Network (CNN). *VisualPhishNet* learns profiles for websites in order to detect phishing websites by a similarity metric that can generalize to pages with new visual appearances. We furthermore present *VisualPhish*, the largest dataset to date that facilitates visual phishing detection in an ecologically valid manner. We show that our method outperforms previous visual similarity phishing detection approaches by a large margin while being robust against a range of evasion attacks.

CCS CONCEPTS

• **Security and privacy → Phishing**; **Web application security**;
• **Computing methodologies → Neural networks**.

KEYWORDS

Phishing Detection; Visual Similarity; Triplet Networks

ACM Reference Format:
Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. 2020. VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3372297.3417233

1 INTRODUCTION

Phishing pages impersonate legitimate websites without permission [49] to steal sensitive data from users causing major financial losses and privacy violations [10, 19, 20, 47]. Phishing attacks have increased due to the advances in creating phishing kits that enabled the deployment of phishing pages on larger scales [10, 37]. According to the Anti-Phishing Working Group (APWG) [2], an international association aiming at fighting phishing attacks, 266,387 attempts have been reported in the third quarter of 2019, which is a high level that has not been witnessed since 2016 [2].

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/3372297.3417233

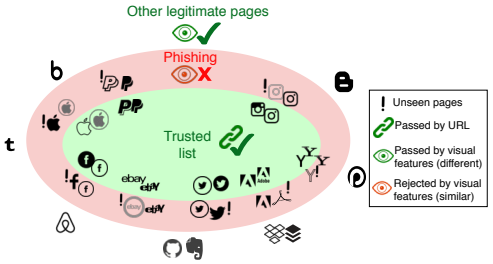


Figure 1: Trusted pages are granted based on their URLs. The remaining pages are compared to the trusted pages by a learnt visual similarity metric. Pages that that are too similar are rejected, which even allows detecting phishing pages with new visual appearances.

There have been numerous attempts to combat the threats imposed by phishing attacks by automatically detecting phishing pages. Modern browsers mostly rely on blocklisting [44] as a fundamentally reactive mechanism. However, in a recent empirical study [36], the new phishing pages that used cloaking techniques were found to be both harder and slower to get detected by blocklists which motivates the development of proactive solutions. An example of the latter is using heuristics that are based on monitored phishing pages [20]. These heuristics can be extracted from URL strings [4, 35, 55] or HTML [9, 26] to detect anomalies between the claimed identity of a webpage and its features [38]. However, since phishing attacks are continuously evolving, these heuristics are subject to continuous change and might not be effective in detecting future attacks [19, 54] (e.g. more than two-thirds of phishing sites in 3Q 2019 used SSL protection [2], its absence formerly was used as a feature to detect phishing pages [38]).

Since the key factor in deceiving users is the high visual similarity between phishing pages and their corresponding legitimate ones, detecting such similarity was used in many previous detection studies [19]. In these methods, a list of commonly attacked pages is maintained, (domain names and screenshots), to protect users from the potential impersonation of such pages; whenever a user visits a page that is not in the trusted-list, its content is compared against the trusted ones. If a high visual similarity is detected, it is then classified as a phishing page as it impersonates one of the trusted pages. Similarity-based methods have the advantage of not relying on heuristics that are likely to evolve and instead they rely on the strong incentive of the adversary to design pages that are similar to trustworthy websites. This makes them less prone to an arms race between defenders and attackers.

These efforts still have limitations. First, their trusted-list are too small in both the number of websites and pages per website (e.g. 4-14 websites in [7, 11, 13, 32, 33], less than 10 pages in [14, 48, 53], 41 pages in [34]) which makes them able to detect attacks against these few pages only. Second, existing approaches fall short in detecting phishing pages that target the same trusted websites but with new unseen visual appearances, as they perform a page-to-page image matching between a previously found phishing page and its legitimate counterpart [5, 14, 19, 25, 39]. Consequently, attackers can bypass detection by using other pages from the targeted websites or by crafting partially similar phishing pages with different background pictures, advertisements, or layout [8, 14, 39].

Contribution. Our work targets the above limitations and focuses on improving and generalizing the image-based visual similarity detection. First, we present *VisualPhish*[1], the **largest dataset to date** (155 trusted websites with 9363 pages), that we constructed **to mitigate the limitations** of previously published datasets, facilitate **visual phishing detection** and improve the ecological validity when evaluating phishing detection frameworks.

Second, we propose *VisualPhishNet*, a similarity-based detection model that is the first to utilize a deep learning approach (in particular, triplet convolutional neural networks) to learn a **robust visual similarity** metric between any two same-website webpages' screenshots, instead of relying on one-to-one matching, which **outperforms prior work by a large margin**. A conceptual overview of our method is depicted in Figure 1; we show a trusted-list of websites in a learnt feature space in which same-website pages have higher proximity. Additionally, phishing webpages have high visual similarity and closer embeddings to the trusted-list, thus, they would be classified as phishing. Contrarily, websites that are outside the list have genuine identities and relatively different features.

2 PRELIMINARIES

In this section, we briefly summarize the related similarity-based phishing detection approaches, then we introduce our threat model.

2.1 Related Work

2.1.1 Page-based similarity approaches. The similarity between phishing and trusted pages can be inferred by comparing HTML features; Huang et al. [18] extracted features that represent the text content and style (e.g. most frequent words, font name and color, etc.), which they used to compare pages against trusted identities. Similarly, Zhang et al. [54] used TF-IDF to find lexical signatures which they used to find the legitimate website domain by a search engine. Besides, Liu et al. [28] segmented a webpage to blocks based on HTML visual cues and compared the layout of two pages by matching blocks. Also, Rosiello et al. [41] used Document Object Model (DOM) comparison, and Mao et al. [33] used Cascading Style Sheet (CSS) comparison. However, these methods fail if attackers used images or embedded objects instead of HTML text [14]. They are also vulnerable to code obfuscation techniques where a different code produces similar rendered images [14, 25].

[1]https://s-abdelnabi.github.io/VisualPhishNet/

2.1.2 Image-based similarity approaches. Consequently, another line of work (which we adopt) infers similarity directly from rendered screenshots. As examples, Fu et al. [14] used Earth Mover's Distance (EMD) to compute the similarity between low-resolution screenshots, which Zhang et al. [53] also used along with textual features. However, this required the images to have the same aspect ratio [25], which is a constraint we do not impose. Also, Lam et al. [25] used layout similarity by matching the screenshots' segmentation blocks. However, the proposed segmentation approach is limited when segmenting pages with complex backgrounds [5]. Our approach does not suffer from these limitations since we use an end-to-end framework to represent images rather than a heuristic-based one. In addition, Chen et al. [8] approximated human perception with Gestalt theory to determine the visual similarity of two webpages' layouts with slight differences (e.g. an addition or removal of a block). They evaluated their approach on only 12-16 legitimate pages and their corresponding spoofed ones. In contrast to these approaches, we generalize the similarity detection and show that our method is not limited to phishing pages with a similar layout to the corresponding trusted ones.

Discriminative keypoint features were often used in phishing detection. As examples, Afroz et al. [1] used Scale-Invariant Feature Transform (SIFT) to match logos, while Rao et al. [39] used Speeded-Up Robust Features (SURF) to match screenshots. Similarly, Bozkir et al. [5] used Histogram of Oriented Gradients (HOG), Chen et al. [7] used Contrast Context Histogram (CCH), and Malisa et al. [31] used Oriented FAST and rotated BRIEF (ORB) to detect mobile applications spoofing. Besides, Medvet et al. [34] used color histograms and 2D Haar wavelet transform of screenshots. However, in recent years, CNNs were shown to significantly outperform local and hand-crafted features in computer vision tasks [23, 43]. Thus, our work is the first to use deep learning in pixel-based visual similarity phishing detection and to study the adversarial perturbations against such models.

Chang et al. [6] and Dunlop et al. [13] used logo extraction to determine a website's identity and then used the Google search engine to find corresponding domains. These approaches assumed a fixed location for the website logo which could be bypassed. Contrary to these approaches, we use a learning-based identification of the discriminating visual cues and study the performance against shifts in location.

Woodbridge et al. [50] used Siamese CNNs to detect visually similar URLs by training on URLs rendered as images. In contrast, we propose a visual similarity metric based on screenshots instead of URL pairs, with further optimizations adapting to the harder problem, which goes beyond homoglyph attacks.

Additionally, despite previous efforts, our work explores new territory in similarity detection research with more generalization and fewer constraints; previous methods aim to form a match between a found phishing attempt and its correspondent real page assuming a highly similar layout and content. Therefore, a phishing page targeting the same website but is different from the trusted pages could go undetected. In addition, same-website pages show a lot of variations in background pictures and colors which attackers might exploit to continuously create new pages. Thus, our model and dataset collection do not rely on page-to-page matching, but on learning a similarity metric between any two same-website pages,

even with different contents, to proactively generalize to partially similar, obfuscated, and unseen pages.

2.2 Threat Model

We consider phishing pages targeting the collected large list of trusted websites. We assume that the attacker would be motivated to target websites that are widely known and trusted, therefore, high coverage of phishing pages could be achieved by the collected trusted-list. We assume that the attacker could craft the phishing page to be fully or partially similar to any page from the targeted websites (not only to pages in the trusted-list), therefore, we relax the page-to-page matching and test on phishing pages that were not seen in the trusted websites' training pages. We study other evasion techniques (hand-crafted and white-box adversarial perturbations) that introduce small imperceptible noise to the phishing page to reduce the similarity to the targeted page that might be contained in the trusted-list. For all these attempts, we assume that the adversary has an incentive to create seemingly trusted pages by not introducing very perceptible noise on the page that might affect the perceived design quality or the website's identity (e.g. large changes to logos and color themes).

3 ANALYSES AND LIMITATIONS OF PUBLISHED DATASETS

In this section, we discuss public datasets and their limitations along with the contributions of the *VisualPhish* dataset.

Unfortunately, only a small number of datasets for the phishing detection task using screenshots are publicly available. One of these is *DeltaPhish* [10] for detecting phishing pages hosted within compromised legitimate websites. The dataset consists of groups having the same domain, where each group contains one phishing page and a few other benign pages from the compromised hosting website. Thus, the legitimate examples only cover the hosting websites, not the websites spoofed by the phishing pages. Consequently, this dataset is not suitable for similarity-based detection. Moreover, we observed that a large percentage of phishing pages' screenshots in this dataset are duplicates since PhishTank[2] reports do not necessarily contain unique screenshots. We also found that the legitimate and phishing examples had different designs as phishing examples generally consisted of login forms with few page elements, while legitimate examples contained more details. This could cause the trained model to be biased to these design changes and, thus, could fail when tested with legitimate pages with login forms.

The Phish-IRIS dataset [11] for similarity-based detection consists of phishing pages collected from PhishTank targeting 14 websites and an "other" class collected from the Alexa top 300 websites[3] representing legitimate examples outside the trusted-list. However, this dataset has a limited number of trusted websites, and the screenshots of the trusted-list were taken only from phishing reports which skews the dataset towards poorly designed phishing pages.

VisualPhish contributions. Based on the previously mentioned limitations, we collected the *VisualPhish* dataset that facilitates similarity-based detection approaches and closes the following gaps: 1) we increased the size of the trusted-list to detect more phishing attacks. 2) we collected a phishing webpage corpus with removing duplicity in screenshots. 3) instead of only training on phishing pages, we also collected legitimate pages of the targeted websites with different page designs and views (i.e. training trusted-list). 4) the dataset is not built on a page-to-page basis but on a per-website basis; the trusted-list contains screenshots from the whole website, phishing pages that target the trusted website are considered even if their counterparts are not found in the trusted-list. 5) we collected a legitimate test set of websites (i.e. different from trusted domains) that limits bias as far as possible (e.g. login forms should also be well represented in this test set).

Unlike previous work, we extend the visual similarity to phishing pages that target the trusted websites but were not seen in the training trusted-list. Thus, we checked that the collected phishing pages are different in terms of simple pixel-wise similarity from the targeted trusted websites' pages. To denote pixel-wise similarity, we used the distances between the pre-trained VGG16 visual representation instead of naive pixel comparison. We computed the minimum distances between the phishing pages and the corresponding targeted website. As a reference, we compared them to the distances between the legitimate test set (other websites) and the trusted-list. If the phishing pages had similar counterparts in the trusted-list, they would have considerably smaller distances compared to other benign pages. However, as can be seen from the two histograms in Figure 2, the distance ranges in both sets are comparable with high overlap. Hence, the phishing pages are different from the training trusted websites' ones and can be used to evaluate the performance on future unseen phishing pages.

4 THE *VISUALPHISH* DATASET

In this section, we show how we constructed *VisualPhish*.

Phishing pages. To collect the phishing examples, we crawled and saved the screenshots of the active verified phishing pages from PhishTank which yielded 10250 pages. We observed that the same phishing screenshot design could be found with multiple URLs, so we manually inspected the saved screenshots to remove duplicates in addition to removing not found and broken URLs. Having non-duplicated screenshots (i.e. unique visual appearance) is important

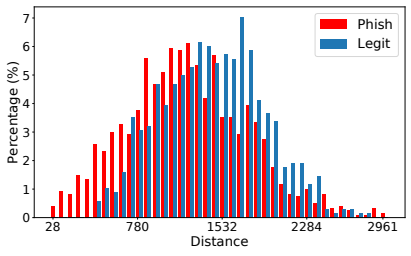


Figure 2: The distances histogram between the pre-trained VGG16 features of the phishing test set and the targeted website in the training trusted-list (red), in comparison with the ones between the benign test set and the trusted-list (blue).

[2]https://www.phishtank.com/
[3]https://www.alexa.com

to have an accurate error estimate and to have a disjoint and non-overlapping training and test splits. After filtering, the phishing set contained 1195 phishing pages targeting 155 websites. We observed that phishing pages targeting one website have differences in elements' locations, colors, scales, text languages and designs (including previous websites' versions), therefore, the phishing set can be used to test the model's robustness to these variations. We also found that some phishing pages are poorly designed with little similarity to the overall design of the targeted website, in addition to having templates that cannot be found in the website but in other applications (e.g. Microsoft Word or Excel files). Such dissimilar examples were excluded from previous work (such as [33]), however, we included all found pages for completeness and to provide a rich dataset for future research. Examples of these variations are in Appendix B. Additionally, the majority of the crawled pages targeted a small subset of the trusted websites (a histogram is in Appendix B), therefore, even though similarity methods cannot detect attacks against non-listed websites, high coverage of phishing pages could be achieved by including a few websites in the trusted-list.

Targeted legitimate websites' pages. Besides collecting phishing webpages, we collected legitimate pages from those 155 targeted websites to work as a visual trusted-list. Instead of only gathering the legitimate counterparts of the found phishing pages as typically done in previous work, we crawled all internal links that were parsed from the HTML file of the homepage. As a result, not all phishing pages have corresponding similar legitimate pages in this trusted-list. We saved all webpages from the website to get different page designs, possible login forms, and different languages to make the similarity model trained with this dataset robust against these differences. For these 155 websites, we collected 9363 screenshots, where the number of collected screenshots for each website depends on the number of hyperlinks found in the homepage.

Top-ranked legitimate websites' pages. Furthermore, we queried the top 500 ranked websites from Alexa, the top 100 websites from SimilarWeb[4], in addition to the top 100 websites in categories most prone to phishing such as banking, finance, and governmental services. In total, we collected a list of 400 websites from SimilarWeb. From these lists, we excluded the 155 websites we collected from the phishing pages' targets, and then we downloaded the screenshots of the top ≈60 websites (non-overlapping) from each list.

Training and test pages split. We have three data components: a training trusted-list of legitimate pages, phishing pages targeting the websites in the trusted-list, and legitimate/benign test examples of websites outside the trusted-list (i.e. different domains). Our objective is to differentiate the phishing pages from other benign examples based on their similarity to the trusted-list.

To train the model, we used the first legitimate set that we built from the phishing pages' targets (155 websites) as a trusted-list that is used only in training. We used a subset of the phishing examples in training as a form of augmentation in order to learn to associate the dissimilar examples to their targets. We do not train on any other legitimate websites (i.e. domains) outside the trusted-list.

To test the model, we used the rest of the phishing set. In addition, we constructed a legitimate test set of 683 benign examples from the top-ranked websites' pages that we crawled (with domains different from the trusted-list); we selected 3-7 screenshots from each website. In order not to have a biased dataset with inherent differences between the legitimate and phishing test sets that might give optimistic or spurious results, we rigorously constructed the legitimate test set such that it contains an adequate number of forms and categories that are used in phishing attacks (e.g. banks, Software as a Service (SaaS), and payment [2]). With a well-balanced test set, we can accurately evaluate the similarity model performance and whether it can find the website identity instead of relying on other unrelated features such as the page layout (e.g. having forms). Additionally, we included other categories (histogram in Appendix B) to have high coverage of websites users might face.

Trusted-list analysis. In addition to the trusted-list we built from PhishTank, we also examined other sources for building trusted-lists without needing to crawl phishing data. This could help in taking proactive steps to protect websites that might be attacked in the future if the adversary decided to avoid detection by targeting other websites than the ones which have been already known to be vulnerable. In order for the attacks to succeed, attackers have an incentive to target websites that are trusted and known for a large percentage of users, therefore, we built our analysis on the top 500 websites from Alexa, and the top 400 websites from SimilarWeb in categories most prone to phishing. To evaluate whether or not these lists can represent the targets that might be susceptible to attacks, we computed the intersection between them and the PhishTank trusted-list. Figure 3 shows cumulative percentages of phishing instances whose targets are included in ascending percentiles of the Alexa, SimilarWeb, and the concatenation of both lists. We found that including both lists covered around 88% of the phishing instances we collected from PhishTank, which indicates that the top-ranked websites are relevant for constructing trusted-lists. Additionally, SimilarWeb list covered more instances than Alexa list, we accounted that for the fact that the former was built from categories such as banks, SaaS and payment, in addition to the general top websites. We, therefore, conclude that this categorization approach is more effective in forming potential trusted-lists since important categories are less likely to change in future attacks.

5 VISUALPHISHNET

As we presented in Figure 1, similarity-based phishing detection is based on whether there is a high visual similarity between a visited

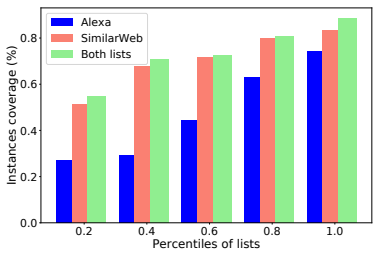


Figure 3: Percentage of phishing instances whose targets are covered by ascending percentiles of other lists.

[4]https://www.similarweb.com/

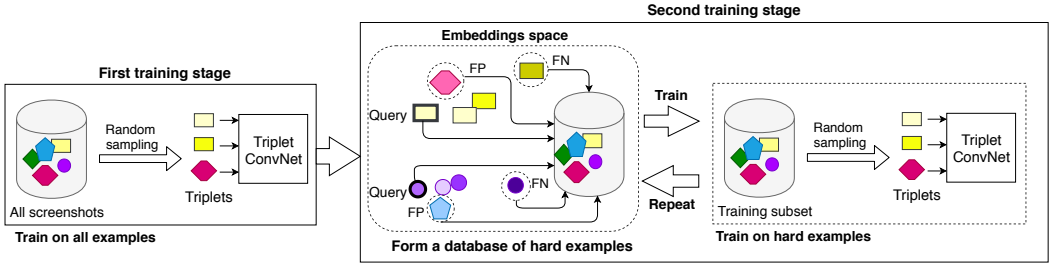


Figure 4: An overview of *VisualPhishNet*. We utilize triplet networks with convolutional sub-networks to learn similarity between same-website screenshots (same shaped symbols), and dissimilarity between different-website screenshots. Our network has two training stages; first, training is performed with uniform random sampling from all trusted-list's screenshots. Second, training is performed by iteratively finding hard examples according to the model's latest checkpoint.

webpage to any of the trusted websites, while having a different domain. If the visited page was found to be not similar enough to the trusted-list, it would be classified as a legitimate page with a genuine identity. Therefore, our objective can be considered as a similarity learning problem rather than a multi-class classification between trusted-list's websites and an "other" class. Including a subset of "other" websites in training with a multi-class classification method could cause the model to fail at test time when testing with new websites. Additionally, instead of the typically used page-to-page correspondence, we aim to learn the similarity between any two same-website pages despite having different contents.

Motivated by these reasons, we treated the problem as a similarity learning problem with deep learning using Siamese or triplet networks which have been successfully used in applications such as face verification [46], signature verification [12], and character recognition [22]. In each of these applications, the identity of an image is compared against a database and the model verifies if this identity is matched with any of those in the database. They have been also used in the tasks of few-shots learning or one-shot learning [22] by learning a good representation that encapsulates the identity with few learning examples. These reasons make this deep learning paradigm suitable for similarity-based phishing detection.

VisualPhishNet, adopts the triplet network paradigm with three shared convolutional networks. We show an overview of the training of *VisualPhishNet* in Figure 4 which consists of two stages: in the first stage, training is performed on all screenshots with a random sampling of examples. The second training stage fine-tunes the model weights by iteratively training on hard examples that were wrongly classified by the model's last checkpoint according to the distance between the embeddings. By learning these deep embeddings, we build a profile for each website that encapsulates its identity, which would enable us to generalize to new webpages that are not contained in the trusted-list database. The rest of this section illustrates in more detail each aspect of the model.

5.1 Triplet Networks

The Siamese networks are two networks with shared weights trained with the goal of learning a feature representation of the input such that similar images have higher proximity in the new feature space than different images. The sub-networks shares weights and parameters and the weight updates are mirrored for each of them, the sub-networks are then joined with a loss function that minimizes the distance of similar objects' embeddings while maximizing the distance of dissimilar objects' ones [12].

The triplet network, which we used in *VisualPhishNet*, extends this approach; it was initially used in the FaceNet system [42] to learn an embedding for the face verification task. This type of architectures performs the training on three images, an anchor image, a positive image whose identity is the same as the anchor, and a negative image with a different identity than the anchor. The overall objective of the network is to learn a feature space in which the distance between the positive and anchor images' embeddings is smaller than the distance between the anchor and negative images' ones. This is achieved by minimizing the loss function that is

$$\text{Loss} = \sum_{i}^{N} \max(\ \|\mathbf{f}(x_i^a) - f(x_i^p)\|_2^2 - \|\mathbf{f}(x_i^a) - f(x_i^n)\|_2^2 + \alpha, 0\ )$$

where: $f(x)$ represents the embedding space (produced by a shared network), (x_i^a, x_i^p, x_i^n) is a set of possible triplets (anchor, positive, and negative), and α is a margin that is enforced between positive and negative pairs which achieves a relative distance constraint. The loss penalizes the triplet examples in which the distance between the anchor and positive images is not smaller by at least the margin α than the distance between the anchor and negative images. In our problem, the positive image is a screenshot of the same website as the sampled anchor, and similarly, the negative image is a screenshot of a website that is different from the anchor.

For the shared network, we used the VGG16 (as a standard architecture) with ImageNet pre-training initialization [45]. We used all layers excluding the top fully connected layers, we then added a new convolution layer of size 5x5 with 512 filters, with ReLU activations, and initialized randomly with HE initialization [16]. Instead of using a fully connected layer after the convolution layers, we used a Global Max Pooling (GMP) layer that better fits the task of detecting possible local discriminating patterns in patches such as logos. To match the VGG image size, all screenshots were resized to 224x224 with the RGB channels.

5.2 Triplet Sampling

Since there are a large number of possible combinations of triplets, the training is usually done based on sampling or mining of triplets

instead of forming all combinations. However, random sampling could produce a large number of triplets that easily satisfy the condition due to having zero or small loss which would not contribute to training. Therefore, mining of hard examples was previously used in FaceNet to speed-up convergence [42].

Therefore, as we show in Figure 4, our training process has two training stages. In the first stage, we used a uniform random sampling of triplets to cover most combinations. After training the network with random sampling, we then fine-tuned the model by iteratively finding the hard examples to form a new training subset. First, we randomly sample a query set representing one screenshot from each website, then with the latest model checkpoint, we compute the L2 distance between the embeddings of the query set and all the rest of training screenshots. In this feature space, the distance between a query image and any screenshot from the same website should ideally be closer than the distance from the same query image to any image from different websites. Based on this, we can find the examples that have the largest error in distance. Hence, we retrieve the one example from the same website that had the largest distance to the query (hard positive example), and the one example from a different website that had the smallest distance to the query (hard negative example). We then form a new training subset by taking the hard examples along with the sampled query set altogether, and we continue the training process with triplet sampling on this new subset. For the same query set, we repeat the process of finding a new subset of hard examples for a defined number of iterations for further fine-tuning. Finally, to avoid overfitting to a query set that might have outliers, we repeat the overall process by sampling a new query set and selecting the training subsets for this new query set accordingly.

This hard example mining framework can be considered as an approximation to a training scheme where a query image is paired with screenshots from all websites and a Softmin function is applied on top of the pairwise distances with a supervised label, however, this would not scale well with the number of websites in the trusted-list, and therefore it is not tractable in our case as a single training example would have 155 pairs (trusted websites).

5.3 Prediction

At test time, the closest screenshot in distance to a phishing test page targeting a website should ideally be a screenshot of the same website. Therefore, the decision is not done based on all triplets comparison but it can be done by finding the screenshot with the minimum distance to the query image. To this end, we use the shared network to compute the embeddings then we compute the L2 distance between the embeddings of the test screenshot and all training screenshots. After computing the pairwise distances, the test screenshot is assigned to the website of the screenshot that has the minimum distance. This step could identify the website targeted if the test page is a phishing page.

As depicted in Figure 1, if the minimum distance between a page and the trusted-list is smaller than a defined threshold, the page would be classified as a phishing page that tries to impersonate one of the trusted websites by having a high visual similarity. On the other hand, if the distance is not small enough, the page would be classified as a legitimate page with a genuine identity. Therefore, we apply a threshold on the minimum distance for classification.

6 EVALUATION

In this section, we first show the implementation details of *VisualPhishNet* and its performance, then we present further experiments to evaluate the robustness of *VisualPhishNet*.

6.1 *VisualPhishNet*: Final Model

Evaluation metrics. Since our method is based on the visual similarity of a phishing page to websites in the trusted-list, we computed the percentage of correct matches between a phishing page and its targeted website. We also calculated the overall accuracy of the binary classification between legitimate test pages and phishing pages at different distance thresholds to calculate the Receiver Operating Characteristic (ROC) curve area.

Implementation details. To train the network, we used Adam optimizer [21] with momentum values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 0.00002 with a decay of 1% every 300 mini-batches where we used a batch size of 32 triplets. We set the margin (α) in the triplet loss to 2.2. The first stage of triplet sampling had 21,000 mini-batches, followed by hard examples fine-tuning, which had 18,000 mini-batches divided as follows: we sampled 75 random query sets, for each, we find a training subset which will be used for 30 minibatches, then we repeat this step 8 times. We used 40% of the phishing examples in training (added to the targeted website pages and used normally in triplet sampling) and used the other 60% for the test set. We used the same training/test split in the two phases of training. We tested the model with the legitimate test set consisting of 683 screenshots; these domains were only used in testing since we train the model on trusted domains only (and partially their spoofed pages).

Performance. Using *VisualPhishNet*, 81% of the phishing test pages were matched to their correct website using the top-1 closest screenshot, while the top-5 match is 88.6%. After computing the correct matches, we computed the false positive and true positive rates at different thresholds (where the positive class is phishing) which yielded a ROC curve area of 0.9879 (at a cut-off of 1% false positives, the partial ROC area is 0.0087) outperforming the examined models and re-implemented visual similarity approaches which we show in the following sections.

6.2 Ablation Study

Given the results of *VisualPhishNet*, this sub-section investigates the effects of different parameters in the model, we summarize our experiments in Table 1 which shows the top-1 match and the ROC area for each model in comparison with the final one (see Appendix A for the ROC curves). We first evaluated the triplet network by experimenting with Siamese network as an alternative. We used a similar architecture to the one used in [22] with two convolutional networks and a supervised label of 1 if the two sampled screenshots are from the same website, and 0 otherwise. The network was then trained with binary cross-entropy loss. We also examined both L1 and L2 as the distance function used in the triplet loss. Besides, we inspected different architecture's parameters regarding the shared

sub-network including the added convolution layer, and the final layer that is used as the embedding vector where we experimented with Global Average Pooling (GAP) [27], fully connected layer, and taking all spatial locations by flattening the final feature map. In addition to VGG16, we evaluated ResNet50 as well [17]. We also studied the effect of the second training phase of hard examples training by comparing it with a model that was only trained by random sampling. As can be seen from Table 1, the triplet network outperformed the Siamese network. Also, the second training phase of hard examples improved the performance, which indicates the importance of this step to reach convergence as previously reported in [42]. We also show that the used parameters in *VisualPhishNet* outperform the other studied parameters. Since some phishing pages had poor quality designs and were different from their targeted websites (see Appendix B for examples), we studied the robustness of *VisualPhishNet* to the ratio of phishing examples seen in training. We, thus, reduced the training phishing set to only 20% and tested with the other 80%, which slightly decreased the top-1 match (mostly on these different examples).

6.3 Trusted-list Expansion

In addition to the PhishTank list gathered from phishing reports, we studied other sources of trusted-lists as per the analysis presented earlier in our dataset collection procedure. We then studied the robustness of *VisualPhishNet*'s performance when adding new websites to the training trusted-list. To that end, we categorized the training websites to three lists (as shown in Figure 5), the PhishTank list, a subset containing 32 websites from SimilarWeb top 400 list (418 screenshots), a subset containing 38 websites (576 screenshots) from Alexa top 500 list. Since we have phishing pages for the websites in the PhishTank list only, the other two lists can be used in training as distractors to the performance on the phishing examples. When training on one of these additional lists, we remove its websites from the legitimate test set yielding test sets of 562 and 573 screenshots in the case of adding SimilarWeb and Alexa lists respectively.

As shown in Table 2, when adding new websites to the training trusted-list, the performance of the classification (indicated by the ROC area and the top-1 match) decreased. However, this decrease in performance was relatively slight, which indicates the robustness of *VisualPhishNet* to adding a few more websites to training.

Sub-network	Added Layer	Last Layer	Network type	Distance	Sampling	%Phishing	Top-1 Match	ROC Area
VGG16	Conv 5x5(512)	GMP	Triplet	L2	2 stages	40%	**81.03%**	**0.9879**
•	•	•	Siamese	•	•	•	75.31%	0.8871
•	•	FC (1024)	Siamese	L1	•	•	64.8%	0.655
•	•	•	•	L1	•	•	73.91%	0.9739
•	•	GAP	•	•	•	•	68.61%	0.6449
•	•	FC (1024)	•	•	•	•	78.94%	0.8517
•	•	Flattening	•	•	•	•	80.05%	0.8721
•	Conv 3x3(512)	•	•	•	•	•	80.19%	0.9174
•	No new layer	•	•	•	•	•	79.91%	0.8703
ResNet50	No new layer	•	•	•	•	•	78.52%	0.8526
•	•	•	•	•	Random	•	75.3%	0.9477
•	•	•	•	•	•	20%	74.37%	0.9899

Table 1: A summary of the ablation study. Row 1 is the finally used model, cells indicated by "•" denotes the same cell value of row 1 (*VisualPhishNet*).

6.4 Comparison with Prior Work and Baselines

Furthermore, we compared *VisualPhishNet* with alternative approaches that we re-implemented on the *VisualPhish* dataset. In recent years, deep learning and CNNs have been demonstrated to achieve a breakthrough over local and hand-crafted features (used in previous work) on many benchmarks [23]. Moreover, off-the-shelf pre-trained CNNs features (even without fine-tuning) have been shown to outperform local features in many tasks [29, 43, 52]. Therefore, we first compare *VisualPhishNet*'s embeddings to the embeddings of two off-the-shelf CNNs: VGG16 and ResNet50. Also, since our work is the first to utilize deep learning, the pre-trained CNNs provide a baseline for deep learning approaches. As we show in Table 3, *VisualPhishNet* outperforms these two baselines with a significant performance gain.

To provide additional evidence, we re-implemented the methods of phishing detection using SURF matching from [39], HOG matching from [5], and ORB matching from [31] which reported that ORB is more suited for the logo detection task than SIFT. Unlike previous work, our approach and dataset do not rely on page-to-page matching, thus, not all phishing pages have legitimate counterparts in the

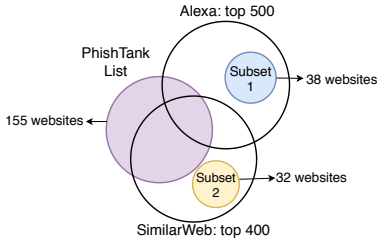


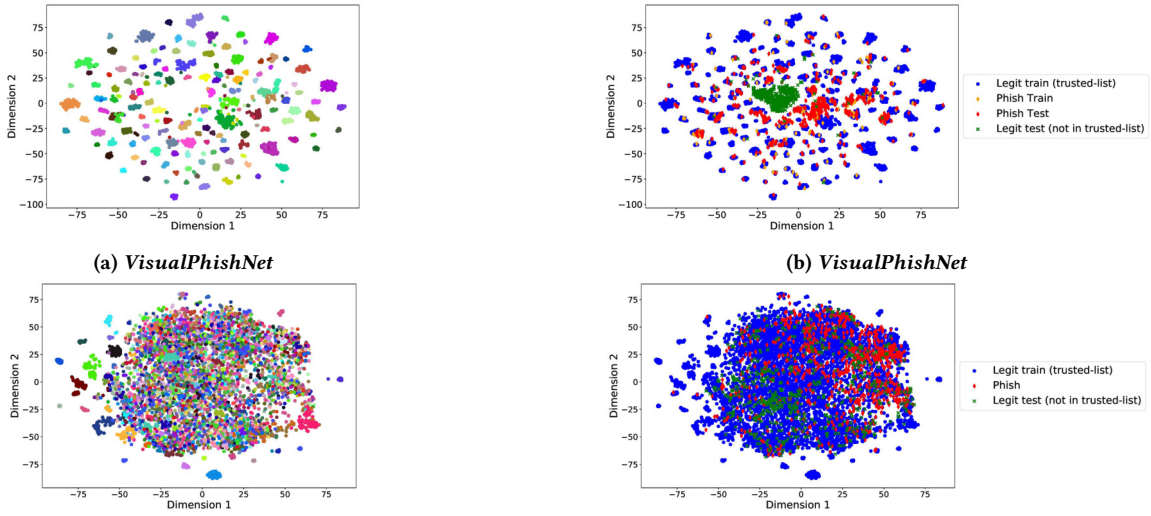
Figure 5: The three main lists used in training, the list collected from PhishTank, a subset of Alexa list, and a subset of SimilarWeb list.

Experiment	Top-1 Match	ROC Area
PhishTank list (155 websites)	81.03%	0.9879
Add SimilarWeb list (32+155 websites)	78.3%	0.9764
Add Alexa list (38+155 websites)	78.1%	0.9681

Table 2: A summary of our experiments when adding more websites from Alexa and SimilarWeb lists to training.

Method	Top-1 Match	ROC Area
VisualPhishNet	**81.03%**	**0.9879**
VGG16	51.32%	0.8134
ResNet50	32.21%	0.7008
ORB	24.9%	0.6922
HOG	27.61%	0.58
SURF	6.55%	0.488

Table 3: Our experiments to compare *VisualPhishNet*'s performance against prior methods and alternative baselines.



(a) *VisualPhishNet* (b) *VisualPhishNet*

(c) VGG16 (d) VGG16

Figure 6: t-SNE visualizations of *VisualPhishNet*'s embeddings compared with the pre-trained VGG16 ones as a baseline. Figures (a) and (c) show the trusted webpages color-coded by websites. Figures (b) and (d) show the trusted webpages (blue) and their phishing pages (red and orange) in comparison with legitimate test pages outside the trusted-lists (green).

training list. This limits the applicability of methods that are based on layout segmentation and explicit block matching (such as [25]). Nevertheless, HOG descriptors, which we compare to, were used to represent the page layout in [5]. As shown in Table 3, the use of pre-trained CNNs (in particular VGG16) does indeed outperform the other baselines. In all of our experiments, similar to *VisualPhishNet* training for a fair comparison, 40% of the phishing set was added to the training list.

This analysis demonstrates that previous image matching methods are not efficient on our dataset containing phishing pages whose contents and visual appearances were not seen in the trusted-list (as shown later in subsection 7.1). Additionally, it shows that pre-trained CNNs are not adequate and further optimization to find the discriminating cues, as done in *VisualPhishNet*, is needed.

6.5 Embeddings Visualization

VisualPhishNet produces a feature vector (dimensions: 512) for each screenshot that represents an encoding that resulted from minimizing the triplet loss. In this learned feature space, same-website screenshots should be in closer proximity compared with screenshots from different websites. To verify this, we used t-Distributed Stochastic Neighbor Embedding (t-SNE) [30] to reduce the dimensions of the embeddings vectors to a two-dimensional set. We show the visualization's results in Figure 6 in which we compare the embeddings of *VisualPhishNet* with pre-trained VGG16 ones (as the best performing baseline). We first visualized the embeddings of the training trusted-list's webpages categorized by websites as demonstrated in Figure 6a and Figure 6c for *VisualPhishNet* and VGG16 respectively. As can be observed, the learned embeddings show higher inter-class separation between websites in the case of *VisualPhishNet* when compared with VGG16. Additionally, Figure 6b and Figure 6d show the training trusted-list's pages in comparison with phishing and legitimate test ones for *VisualPhishNet* and VGG16 respectively. For successful phishing detection, phishing pages should have smaller distances to trusted-list's pages than legitimate test pages, which is more satisfied in the case of *VisualPhishNet* than VGG16.

6.6 Distance Threshold Selection

To determine a suitable distance/similarity threshold for the binary classification between phishing and legitimate test sets, we split the phishing and legitimate hold-out sets to validation and test sets. We computed the minimum distances of both of them to the training trusted-list. Figure 7a shows the two density histograms and the fitted Gaussian Probability Density Functions (PDF) of the minimum distance for the validation sets of both classes. The vertical line (at ≈8) represents a threshold value with an equal error rate. Additionally, Figure 7b shows the true and false positive rates of the test sets over different thresholds where the indicated threshold is the same one deduced from Figure 7a, which achieves ≈93% true positive rate at ≈4% false positive rate.

6.7 Robustness and Security Evaluation

To test the robustness of *VisualPhishNet*, we define two models for evasion attacks. First, we study how susceptible *VisualPhishNet* is to small changes in the input (e.g. change of color, noise, and position). Second, we assume a white-box attack where the adversary has full access to the target model and the dataset used in training (including the closest point to the phishing page). In both models, we assume that the attacker's goal is to violate the target model's integrity (in our case: similarity detection to the targeted website) by crafting phishing pages that show differences from their corresponding original pages that might be included in the trusted-list. However,

we assume that the adversary is motivated to not introduce very perceivable degradation in the design quality for his phishing page to seem trusted and succeed in luring users.

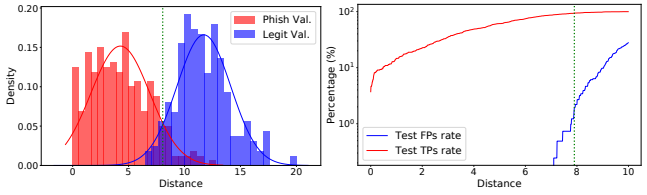
Performance against hand-crafted perturbations. We studied 7 types of perturbations [51] that we applied to the phishing test set (without retraining or data augmentation): blurring, brightening, darkening, Gaussian noise, salt and pepper noise, occlusion by insertion of boxes, and shifting. Table 4 demonstrates an example of each of these changes along with the corresponding relative decrease in performance. Our findings revealed that the matching accuracy and the ROC area dropped slightly (by up to ≈4.3% and ≈1.8% respectively) for the imperceptible noise, while it dropped by up to ≈6.7% and ≈5% respectively for the stronger noise that we assume that it is less likely to be used. Further improvement could be achieved with data augmentation during training.

Adversarial perturbations. Another direction for evasion attacks is crafting adversarial perturbations with imperceptible noise that would change the model decision when added to the input test points [24]. There is a lot of work towards fixing the evasion problem [3], however, adversarial perturbations are well-known for classification models. In contrast, *VisualPhishNet* is based on a metric learning approach that, at test time, is used to compute distances to the training points. We are not aware of any prior adversarial perturbation methods on similarity-based networks and therefore we propose and investigate an adaptation of the adversarial example generation methods to our problem by using the Fast Gradient Sign Method (FGSM) [15] defined as:

$$\widetilde{x} = x + \epsilon \ \text{sign}(\nabla_x J(\theta, x, y))$$

where $\widetilde{x}$ is the adversarial example, x is the original example, y is the example's target (0 in the triplet loss), θ denotes the model's parameters and J is the cost function used in training (triplet loss in *VisualPhishNet*). Adapting this to our system, we used the phishing test example as the anchor image, sampled an image from the same website as the positive image (from the training trusted-list), and an image from a different website as the negative image. We then computed the gradient with respect to the anchor image (the phishing test image) to produce the adversarial example. We experimented with two values for the noise magnitude (ϵ): 0.005 and 0.01. The 0.01 noise value is no longer imperceptible and causes noticeable noise in the input (as shown in Figure 8). We also examined different triplet sampling approaches when generating the adversarial examples, in the first one, we select the positive image randomly from the website's images. However, since the matching decision is based on the closest distance, in the second approach, we select the closest point as the positive. We demonstrate our results in Table 5 where we show the relative decrease in the top-1 matching accuracy and the ROC AUC for each case averaged over 5 trials as we randomly sample triplets for each example. Our results showed that the matching accuracy and the AUC dropped by ≈10.5% and ≈6.5% for the 0.005 noise and by ≈22.8% and ≈12.4% for the higher 0.01 noise. Also, targeting the closest example was similar to sampling a random positive image. Besides, we tested an iterative approach of adding a smaller magnitude of noise to the closest point at each step (0.002 noise magnitude for 5 steps) which was comparable to adding noise with a larger magnitude (0.01) at only one step.

We then performed adversarial training by fine-tuning the trained *VisualPhishNet* for 3000 mini-batches. In each mini-batch, half of the triplets were adversarial examples generated with FGSM with an epsilon value that is randomly generated from a range of 0.003 and 0.01. After training, we again applied FGSM on the phishing test set using the tuned model. As shown in the last two columns of Table 5, the performance improved to reach a comparable performance to the original set in the case of the 0.005 noise. These

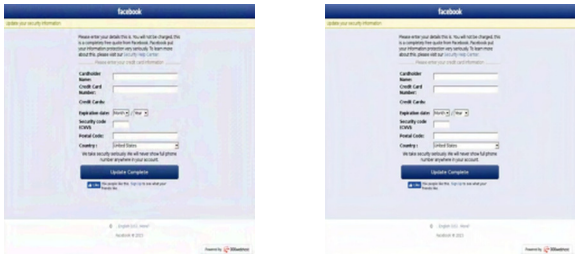


(a) (b)

Figure 7: Distance threshold selection. (a) shows a density histogram of the minimum distances between the phishing (red) and legitimate (blue) validation sets to the training trusted-list. (b) shows the true and false positive rates of the test sets over thresholds, the vertical green line marks the threshold from (a).

	Blurring	Darkening	Brightening	Gaussian noise	Salt and Pepper	Occlusion	Shift
	Sigma=1.5	Gamma=1.3	Gamma=0.8	Var=0.01	Noise=5%	Last quarter	(-30,-30) pixels
							
Matching drop	1.38%	4.31%	1.72%	1.9%	2.07%	1.2%	3.09%
ROC AUC drop	0.17%	1.56%	0.36%	1.47%	1.79%	0.12%	0.86%
	Sigma=3.5	Gamma=1.5	Gamma=0.5	Var=0.1	Noise=15%	Second quarter	(-50,-50) pixels
							
Matching drop	4.13%	5.68%	6.36%	6.71%	6.54%	5.34%	6.54%
ROC AUC drop	1.17%	2.65%	3.35%	2.65%	3.04%	4.99%	1.65%

Table 4: The studied hand-crafted perturbations applied to the phishing test set. The table shows the relative decrease in the top-1 matching accuracy and ROC AUC with respect to the performance on the original phishing set.



(a) (b)

Figure 8: Adversarial examples generated with FGSM on the triplet loss with $\epsilon = 0.01$ (a) and $\epsilon = 0.005$ (b).

results demonstrate that *VisualPhishNet*, after retraining, is robust against adversarial attacks with slightly added noise.

Evaluating different browsers. We studied the effect of the changes caused by other browsers than the one we used to build the dataset (Firefox) as an example of one of the factors that could be different when deploying the system. Thus, we created a subset of 50 URLs from 14 websites, and we used Firefox, Opera, Google Chrome, Microsoft Edge, and Vivaldi browsers to take screenshots of these pages of which we computed the *VisualPhishNet*'s embeddings. In Table 6, we quantify the browsers' changes by comparing the L2 differences between Firefox's embeddings (to match the dataset) and other browsers' ones, which we found smaller by at least $\approx 6.6x$ than the differences caused by the slight hand-crafted perturbations (applied on Firefox screenshots) we previously showed in the first row of Table 4 and demonstrated that they already had a small effect on the performance. Additionally, some of these browser differences were due to advertisement or color differences which are already included in the constructed dataset (see Appendix B).

6.8 Testing with New Crawled Data

Zero-day pages. To provide additional evidence for the efficacy of *VisualPhishNet* in detecting zero-day pages, we crawled recent 955 PhishTank pages targeting the trusted-list (examples in Appendix B). These are new pages that were created and captured after dataset collection, training, and evaluating the model with all previous experiments, and therefore, they are future pages with temporal separation with respect to the model. Additionally, We used a different browser, machine, and screen size from the ones used to collect the dataset to further test against possible variations. We then tested the trained model with this new set (without retraining), and 93.25% were correctly matched (top-5: 96%), compared to 81% (top-5: 88%) on the harder and more dissimilar dataset's phishing pages (see Appendix A for matching examples). Additionally, as a baseline, the VGG-16 matching accuracy of this new set is 65.8%.

Alexa top-10K. To further test the false positives, we crawled the Alexa top 10K websites (excluding the trust-list's domains) to use as a benign test set. Using the same trained model, the ROC AUC of classifying this new benign set against the original phishing set is 0.974, while the partial ROC AUC at 1% false positives is 0.0079 (compared to 0.987 and 0.0087, respectively, on the smaller benign subset). Similarly, the corresponding VGG-16 ROC AUC is 0.781.

Model	Epsilon (ϵ)	Sampling	Matching drop	ROC AUC drop
Original	0.005	random	10.5%	6.47%
	0.005	closest point	10.11%	6.07%
	0.01	random	22.81%	12.35%
	0.002	iterative	20.8%	12.05%
Retrained	0.005	random	**2.54%**	**0.07%**
	0.01	random	9.78%	3.61%

Table 5: The relative performance decrease (with respect to the original test set) of the FGSM adversarial examples.

Browser	Chrome	Edge	Opera	Vivaldi
	0.278±0.54	0.23±0.75	0.271±0.21	0.41±0.57
Noise	Blurring	Gaussian	Salt and Pepper	Shift
	4.92±2.71	2.73±1.02	6.80±2.24	5.43±2.36

Table 6: The L2 difference between Firefox screenshots' embeddings and other browsers' ones, compared to the L2 difference due to the studied slight perturbations.

7 DISCUSSION

We discuss the implications of the efficacy of *VisualPhishNet* by showcasing phishing examples that were correctly detected, and failure modes with both false positive and false negative examples.

7.1 Evaluating Successful Cases

We categorize the successfully classified phishing pages into three main categories. The first one is the easily classified ones consisting of exact or very close copying of a corresponding legitimate webpage from the training trusted-list. However, our model still showed robustness to small variations such as the text language of login forms (which shows an advantage over text-similarity methods), small advertisements' images changes, the addition or removal of elements in the page, and changes in their locations. We observed that these pages have approximately a minimum distance in the range of 0-2 to the training set (as shown in the distances' histogram in Figure 7) and constitute around 25% of the correct matches. The second category, which is relatively harder than the first one, is the phishing webpages that look similar in style (e.g. location of elements and layout of the page) to training pages, however, they are

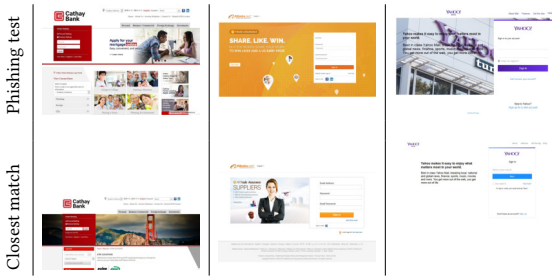


Figure 9: Test phishing pages (first row) that were correctly matched to the targeted websites (closest match from the training set in the second row) with the closest pages having a relatively similar layout but different colors and content.

highly different in content (e.g. images, colors, and text). We show examples of this second category in Figure 9. Similarly, these pages correspond approximately to the distance range of 2-4 in Figure 7 and constitute around 35% of the correct matches.

Finally, the hardest category is the phishing pages showing disparities in design when compared to the training examples as shown in Figure 10. These pages had distances to the training set which were higher than 4 and increased according to their differences and they constitute around 40% of the correct matches. For example, the first three columns show a match between pages with different designs and elements' locations. Also, the fourth phishing page has a pop-up window that partially occludes information and changes the page's colors. The fifth phishing page is challenging as it does not show the company logo, yet it was correctly matched to the targeted website due to having other similar features. This suggests that *VisualPhishNet* captures the look and feel of websites, which makes it have an advantage over previous matching methods that relied only on logo matching such as [1, 13]. The last two pages are highly dissimilar to the matched page except for having the same logo and other similar colors. Even though these examples could arguably be easily recognized as phishing pages by users, they are more challenging to be detected based on similarity and therefore they were excluded in previous studies such as [33], however, we included them for completeness. This analysis shows the ability of *VisualPhishNet* to detect the similarity of phishing pages that are partially copied or created with poor quality in addition to unseen phishing pages with no counterparts in the training trusted-list, which all are possible attempts to evade detection in addition to the ones we previously discussed. We also show in Appendix A phishing examples targeting different websites that have highly similar colors but they were correctly distinguished from each other.

Since these successful matches suggest that the logo of a page plays an important factor in the matching decision, possible false matches could happen if a benign page contains another website logo. To evaluate this, we collected a benign subset of 125 pages (see Appendix B) that contain the logos of one or more of 9 trusted websites. These pages are articles about a website, or login pages with other websites' sign-in or sharing options. However, only 3.07% of these pages were matched to the website whose logo appears in the screenshot which indicates that the learnt profiles incorporate more visual cues than logos only.

7.2 Evaluating Failure Modes

We also analysed the failure modes of the model including wrong websites matches and false positives. We found that the highest mismatches are for phishing examples belonging to Facebook, Dropbox, Microsoft one drive, Microsoft Office, and Adobe. We found that these websites have many phishing pages with dissimilar appearances (and poor designs) compared to the targeted websites, such as the first three phishing pages targeting Facebook and Microsoft Excel in Figure 11 (see also Appendix B for more examples). On the other hand, phishing pages targeting banks had higher quality in copying and appeared plausible and similar to the targeted websites making them have fewer mismatches (see Appendix A for a histogram of wrong matches). To analyse how successful these dissimilar pages in fooling users, we conducted an online study where users were shown dissimilar and relatively similar phishing pages and were asked to evaluate how trustworthy they seem based only on their appearance. Only 3.02% said that they would trust the dissimilar examples as opposed to 65.3% in the case of the relatively similar ones (see Appendix B for examples used in the study).

We also found some phishing pages that used outdated designs or earlier versions of certain login forms such as the fourth example in Figure 11 (that is now changed entirely in Microsoft website) and were, therefore, matched to a wrong website. This could be improved by including earlier versions of websites in the training data. Moreover, the last three examples in Figure 11 show some of the main limitations. Since our training trust-list contains a large number of screenshots per website, we have many distractors of potentially similar pages to the query screenshot, such as the fifth and sixth examples in Figure 11 that were matched to similar screenshots from different websites. We also found that some phishing pages have pop-up windows that completely covered the logo and the page's colors and structure, and were then matched to pages with darker colors such as the last example in Figure 11. The wrong matches had generally higher distances than the correct matches which could make them falsely classified as legitimate examples.

We also show false positive examples (benign test pages) that had high similarity to pages from the training set in Figure 12 and would be falsely classified as phishing pages based on the threshold in Figure 7. We observed that pages with forms were harder to identify as dissimilar to other pages with forms in the trust-list especially when having similar colors and layout, since they contain few distinguishable and salient elements and they are otherwise

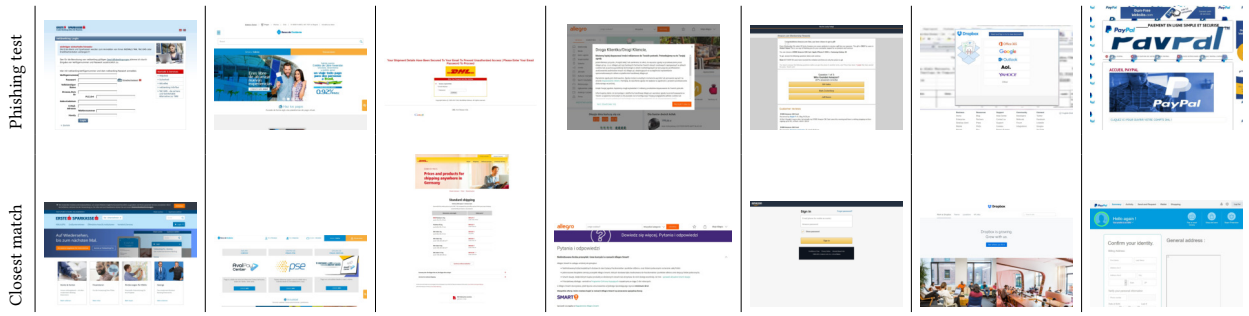


Figure 10: Examples of test phishing webpages that were correctly matched to the targeted websites despite having large differences in layout and content.

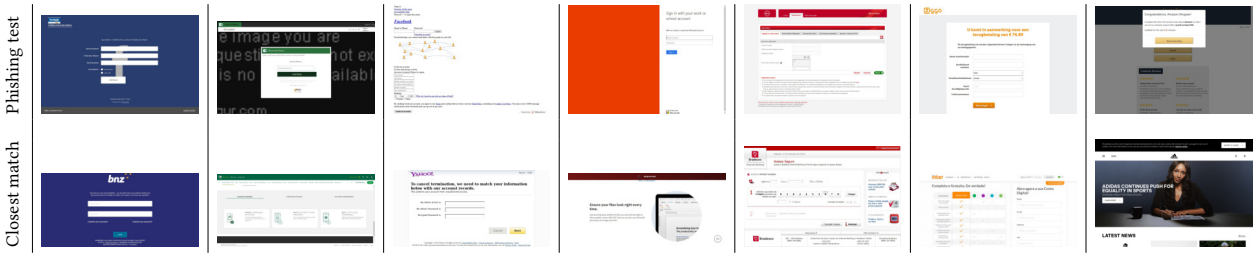


Figure 11: Examples of test phishing webpages that were matched to the wrong website from the training set.

similar. We believe that using the screenshot's text (possibly extracted by OCR), or more incorporation of the logo features along with other visual cues by region-based convolution [40] could be future possible model optimization directions to help reduce the false positives and also improve the matching of hard examples. Additionally, tackling the phishing problem has many orthogonal aspects; while we focus on visual similarity to detect zero-day and unseen pages and achieve a significant leap in performance, our approach could still be used along with other allow-listing of trusted domains to further reduce the false positives.

7.3 Deployment Considerations

We here discuss practical considerations for the deployment of our system. First, regarding the required storage space and computation time, our system does not require storing all screenshots of the trusted-list, as it suffices to store the embedding vectors of screenshots (512-dimensional vectors). Also, the system is computationally feasible since the training trusted-list embeddings can be pre-computed, which at test time only leaves the relatively smaller computations of the query image embedding and the L2 distances. On a typical computer with 8 GByte RAM and Intel Core i7-8565U 1.80GHz processor, the average time for prediction was 1.1±0.7 seconds which decreased to 0.46±0.25 seconds on a NVIDIA Tesla K80 GPU. If further speeding up is needed, the search for the closest point could be optimized. Besides, the decision could only be computed when the user attempts to submit information. We also show in our analysis of possible perturbations that the learned similarity is robust against partial removal of parts of the page, which suggests that a page could be detected even if it was partially loaded. Other deployment issues are the browser window size variations at test time which could be solved by fixing the size of the captured screenshot. Another issue is the maintenance of the domain names of the trusted-list in case a website has changed its domain, which could be solved by rolling updates of the trusted-list without the need to retrain. Additionally, we observed that *VisualPhishNet* is robust against small changes or updates in the website logo's fonts or colors (e.g. see Yahoo examples with different versions that were still correctly detected in Appendix A). Larger or more significant changes (that usually happen on long time intervals) might require retraining and updating. Moreover, the current system and dataset are focusing on Desktop browsers, however, the concept can be extended to other devices (e.g. smartphones) which may require re-training. Furthermore, our visual similarity model can either be used as a standalone phishing detection model or, as the last defense mechanism for unseen pages along with other (potentially faster) listing or heuristics approaches. Regarding the *VisualPhish* dataset, we point out that the manual work in curating the dataset was mainly for constructing unbiased and non-duplicated test sets, however, it is less needed in collecting the training trusted-list of trusted websites. This enables the automatic update of the trusted-list to add new websites when needed. Nevertheless, detecting duplicity can be automated by finding the closest pages to the newly added one based on pixel-wise features (such as VGG features).

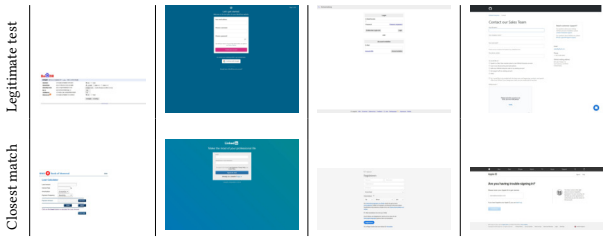


Figure 12: False positive examples of the top closest legitimate test pages to the training list.

8 CONCLUSION

As visual similarity is a key factor in detecting zero-day phishing pages, in this work, we proposed a new framework for visual similarity phishing detection. We presented a new dataset (*VisualPhish*: 155 websites with 9363 screenshots) that covers the largest trusted-list so far and overcomes the observed previous limitation.

Unlike previous work, instead of only matching a phishing page to its legitimate counterpart, we generalize visual similarity to detect unseen pages targeting the trusted websites. To that end, we proposed *VisualPhishNet* that learns a visual profile of websites by learning a similarity metric between any two same-website pages despite having different contents. Based on the qualitative analysis of the successful cases, our network identified easy phishing pages (highly similar to pages in training), and importantly, phishing pages that were partially copied, obfuscated, or unseen. *VisualPhishNet* is robust against the range of possible evasion attacks and perturbations that we studied, which makes our model less prone to the fierce arms race between attackers and defenders.

In conclusion, our work introduces important contributions to phishing detection research to learn a robust and proactive visual similarity metric that demonstrates a leap in performance over prior visual similarity approaches by 56 percent points in matching accuracy and 30 in the classification ROC area under the curve.

REFERENCES

[1] Sadia Afroz and Rachel Greenstadt. 2011. Phishzoo: Detecting phishing websites by looking at them. In *Proceedings of the IEEE International Conference on Semantic Computing*.

[2] APWG. 2019. Anti Phishing Working Group report. https://www.antiphishing.org/resources/apwg-reports/.

[3] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84 (2018), 317–331.

[4] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. 2010. Lexical feature based phishing URL detection using online learning. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*.

[5] Ahmet Selman Bozkir and Ebru Akcapinar Sezer. 2016. Use of HOG descriptors in phishing detection. In *Proceedings of the IEEE International Symposium on Digital Forensic and Security (ISDFS)*.

[6] Ee Hung Chang, Kang Leng Chiew, Wei King Tiong, et al. 2013. Phishing detection via identification of website identity. In *Proceedings of the IEEE International Conference on IT Convergence and Security (ICITCS)*.

[7] Kuan-Ta Chen, Jau-Yuan Chen, Chun-Rong Huang, and Chu-Song Chen. 2009. Fighting phishing with discriminative keypoint features. *IEEE Internet Computing* 13, 3 (2009), 56–63.

[8] Teh-Chung Chen, Scott Dick, and James Miller. 2010. Detecting visually similar web pages: Application to phishing detection. *ACM Transactions on Internet Technology (TOIT)* 10, 2 (2010), 5.

[9] Neil Chou, Robert Ledesma, Yuka Teraguchi, Dan Boneh, and John C. Mitchell. 2004. Client-side defense against web-based identity theft. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[10] Igino Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli. 2017. Deltaphish: Detecting phishing webpages in compromised websites. In *Proceedings of European Symposium on Research in Computer Security (ESORICS)*. Springer.

[11] Firat Coskun Dalgic, Ahmet Selman Bozkir, and Murat Aydos. 2018. Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors. In *Proceedings of the IEEE International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*.

[12] Sounak Dey, Anjan Dutta, J Ignacio Toledo, Suman K Ghosh, Josep Lladós, and Umapada Pal. 2017. Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131* (2017).

[13] Matthew Dunlop, Stephen Groat, and David Shelly. 2010. Goldphish: Using images for content-based phishing analysis. In *Proceedings of the IEEE International Conference on Internet Monitoring and Protection*.

[14] Anthony Y Fu, Liu Wenyin, and Xiaotie Deng. 2006. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). *IEEE Transactions on Dependable and Secure Computing* 3, 4 (2006), 301–311.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Chun-Ying Huang, Shang-Pin Ma, Wei-Lin Yeh, Chia-Yi Lin, and Chien-Tsung Liu. 2010. Mitigate web phishing using site signatures. In *Proceedings of the IEEE Region 10 Conference (TENCON)*.

[19] Ankit Kumar Jain and B Brij Gupta. 2017. Phishing detection: analysis of visual similarity based approaches. *Security and Communication Networks* (2017).

[20] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. 2013. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials* 15, 4 (2013), 2091–2121.

[21] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[22] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *International Conference on Machine Learning (ICML) Deep Learning Workshop*.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

[24] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*.

[25] Ieng-Fat Lam, Wei-Cheng Xiao, Szu-Chi Wang, and Kuan-Ta Chen. 2009. Counteracting phishing page polymorphism: An image layout analysis approach. In *Proceedings of the International Conference and Workshops on Advances in Information Security and Assurance*. Springer.

[26] Yukun Li, Zhenguo Yang, Xu Chen, Huaping Yuan, and Wenyin Liu. 2019. A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems* 94 (2019), 27–39.

[27] Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network in network. In *International Conference on Learning Representations (ICLR)*.

[28] Wenyin Liu, Xiaotie Deng, Guanglin Huang, and Anthony Y Fu. 2006. An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing* 10, 2 (2006), 58–65.

[29] Jonathan L Long, Ning Zhang, and Trevor Darrell. 2014. Do convnets learn correspondence?. In *Advances in Neural Information Processing Systems*.

[30] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[31] Luka Malisa, Kari Kostiainen, and Srdjan Capkun. 2017. Detecting mobile application spoofing attacks by leveraging user visual similarity perception. In *Proceedings of the ACM on Conference on Data and Application Security and Privacy*.

[32] Jian Mao, Pei Li, Kun Li, Tao Wei, and Zhenkai Liang. 2013. BaitAlarm: detecting phishing sites using similarity in fundamental visual features. In *Proceedings of the IEEE International Conference on Intelligent Networking and Collaborative Systems*.

[33] Jian Mao, Wenqian Tian, Pei Li, Tao Wei, and Zhenkai Liang. 2017. Phishing-alarm: robust and efficient phishing detection via page component similarity. *IEEE Access* 5 (2017), 17020–17030.

[34] Eric Medvet, Engin Kirda, and Christopher Kruegel. 2008. Visual-similarity-based phishing detection. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*.

[35] Luong Anh Tuan Nguyen, Ba Lam To, Huu Khuong Nguyen, and Minh Hoang Nguyen. 2014. A novel approach for phishing detection using URL-based heuristic. In *Proceedings of the IEEE International Conference on Computing, Management and Telecommunications (ComManTel)*.

[36] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. 2019. 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 (SP)*.

[37] Adam Oest, Yeganeh Safei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Gary Warner. 2018. Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *APWG Symposium on Electronic Crime Research (eCrime)*.

[38] Ying Pan and Xuhua Ding. 2006. Anomaly based web phishing page detection. In *Proceedings of the IEEE Annual Computer Security Applications Conference (ACSAC)*.

[39] Routhu Srinivasa Rao and Syed Taqi Ali. 2015. A computer vision technique to detect phishing attacks. In *Proceedings of the IEEE International Conference on Communication Systems and Network Technologies*.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*.

[41] Angelo PE Rosiello, Engin Kirda, Fabrizio Ferrandi, et al. 2007. A layout-similarity-based approach for detecting phishing pages. In *Proceedings of the IEEE International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm)*.

[42] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[43] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshops*.

[44] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang. 2009. An empirical analysis of phishing blacklists. In *the Sixth Conference on Email and Anti-Spam (CEAS)*.

[45] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*.

[46] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[47] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, et al. 2017. Data breaches, phishing, or malware?: Understanding the risks of stolen credentials. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

[48] Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Zhang Min, and Xiaotie Deng. 2005. Detection of phishing webpages based on visual similarity. In *Special interest tracks and posters of the 14th international conference on World Wide Web*.

[49] Colin Whittaker, Brian Ryner, and Marria Nazif. 2010. Large-Scale Automatic Classification of Phishing Pages. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[50] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. 2018. Detecting Homoglyph Attacks with a Siamese Neural Network. In *Proceedings of the IEEE Security and Privacy Workshops*.

[51] Ning Yu, Larry Davis, and Mario Fritz. 2019. Attributing fake images to GANs: learning and analyzing GAN fingerprints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.
[52] Joe Yue-Hei Ng, Fan Yang, and Larry S Davis. 2015. Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshops*.
[53] Haijun Zhang, Gang Liu, Tommy WS Chow, and Wenyin Liu. 2011. Textual and visual content-based anti-phishing: a Bayesian approach. *IEEE Transactions on Neural Networks* 22, 10 (2011), 1532–1546.
[54] Yue Zhang, Jason I Hong, and Lorrie F Cranor. 2007. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*.
[55] Mouad Zouina and Benaceur Outtaj. 2017. A novel lightweight URL phishing detection system using SVM and similarity index. *Human-centric Computing and Information Sciences* 7, 1 (2017), 98.

A EXTRA EVALUATION AND QUALITATIVE RESULTS

We here show supplementary results. In Figure 13, we present the ROC curves of the binary classification task for each experiment in the ablation study (discussed in subsection 6.2). In Figure 14, we show a histogram of the phishing pages false matches per website (subsection 7.2). In Figure 15, we show successful matching examples of the new crawled phishing pages (subsection 6.8). Moreover, in Figure 16, we show correct matches across different websites with similar colors that were still correctly distinguished from each other (subsection 7.1). Finally, in Figure 17, we show successful examples where the website logo's colors and fonts were different than the trusted-list to test versions changes (subsection 7.3).

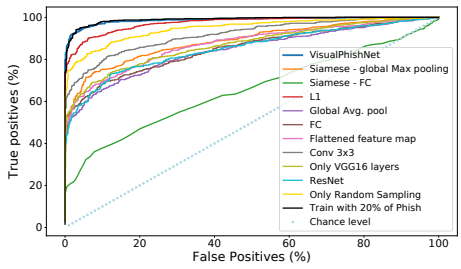


Figure 13: ROC curves for the ablation study in Table 1. The legend follows the same order of rows in Table 1.

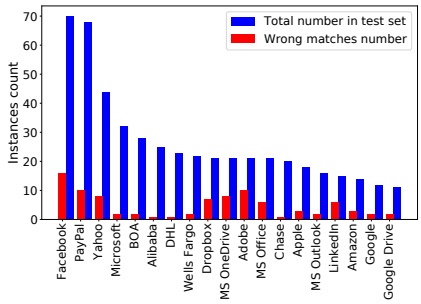


Figure 14: Histogram of the wrong matches of phishing pages to their targeted website. The most frequent 19 websites are shown.

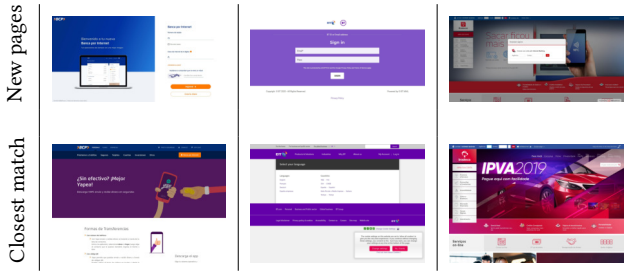


Figure 15: Examples of the newly crawled phishing pages (row 1) that were correctly matched to the targeted website where the closest trusted-list screenshots are in row 2.

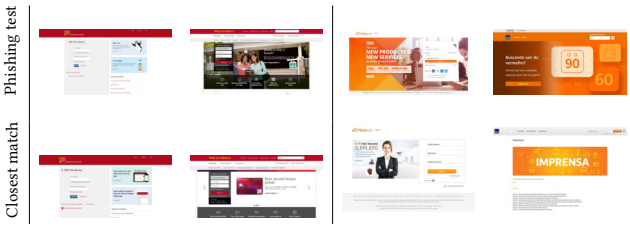


Figure 16: Examples of websites with similar colors (Wells Fargo vs. CIBC, Alibaba vs. Banco Itaú) that were correctly distinguished from each other.

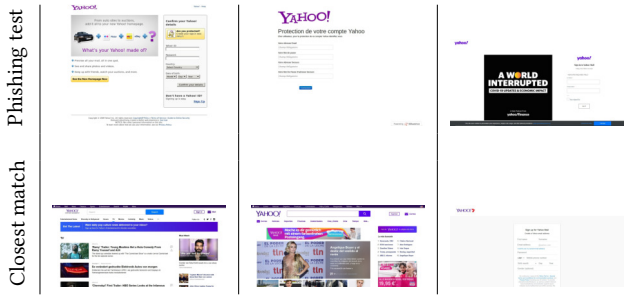


Figure 17: Examples of successfully matched phishing pages where the website logo's fonts and colors are slightly different than the trusted-list. The first two examples contain an older version's logo, while the third example contains a newer version's logo (from the newly crawled data).

B MORE DATASET DETAILS AND EXAMPLES

We show here more details about the *VisualPhish* dataset. In Figure 18, we show a histogram of the most targeted websites by the crawled phishing pages (section 4). Figure 19 shows the categories in the benign test set that we constructed to reduce bias by having similar categories to the trusted websites (section 4). Figure 20 shows examples of the test set used to test browser differences (subsection 6.7). Figure 21 shows examples of the test set used to test false positives when trusted logos are found in the page (subsection 7.1). Examples of the variations (e.g. designs, colors and layout) of the dataset's phishing pages targeting one website are demonstrated in Figure 22. Examples of the poorly designed (i.e. dissimilar to their targets) phishing pages are in Figure 23. Also, Figure 24 shows examples of the screenshots used in the online user study to evaluate dissimilar examples (discussed in subsection 7.2). Finally, Figure 25 shows examples of the newly crawled phishing pages to test the performance against zero-day pages (subsection 6.8).

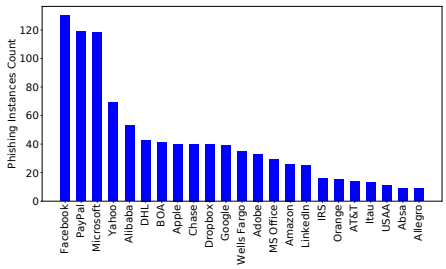


Figure 18: A histogram of the 23 most frequent websites that were found in the unique phishing set.

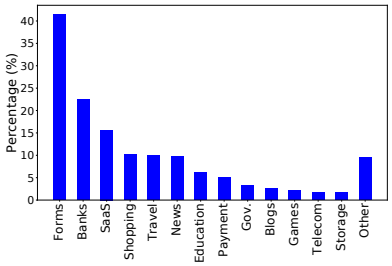


Figure 19: The categories in the legitimate test set.

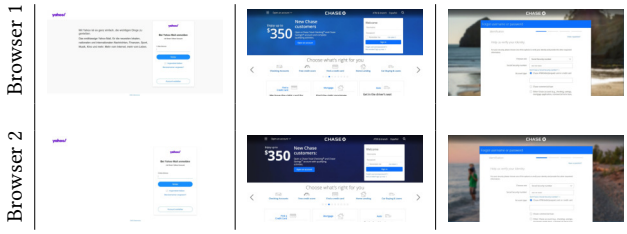
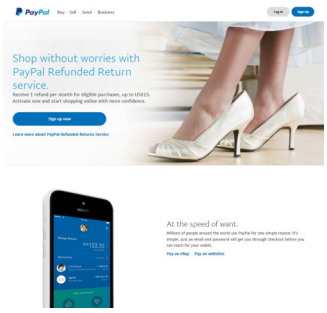
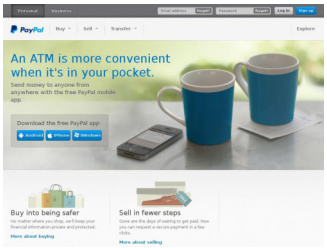


Figure 20: Examples of the differences found between different browsers from the 50 pages used to evaluate the effect of browsers differences.

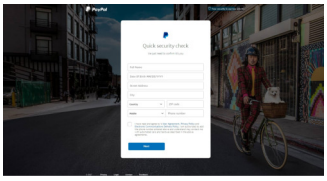
Figure 21: Examples of the test set (consisting of 125 pages) used to evaluate the possible wrong matching to a trust-list's website whose logo appears in other benign pages (such as articles and login pages).



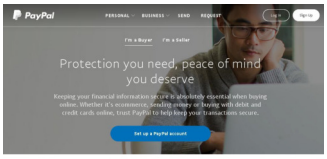
(a)



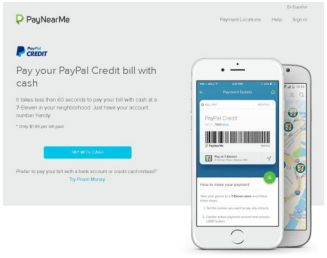
(b)



(c)



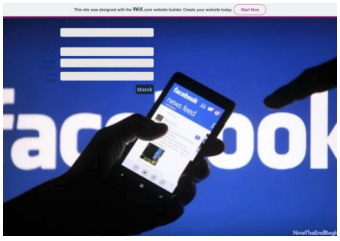
(d)



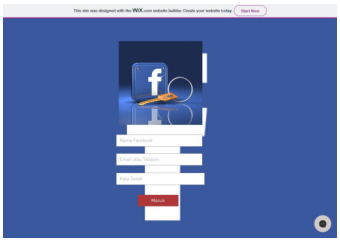
(e)

Figure 22: Examples of the variations in the *VisualPhish* dataset of phishing examples targeting one website with no counterparts in the crawled legitimate examples (training list) of the same website.

(a)

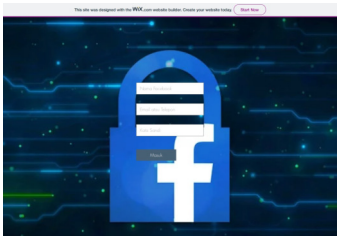


(b)



(c)

(d)



(e)

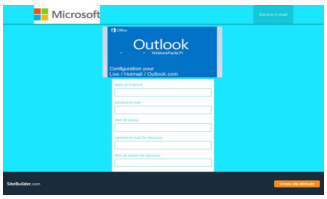
Figure 23: Examples of phishing pages in the dataset that are not similar enough (either in colors or design) to the legitimate website which causes an increase in the mismatches when not partially train with a part of the phishing set.

(a)

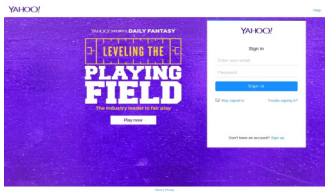
(b)

(c)

(d)

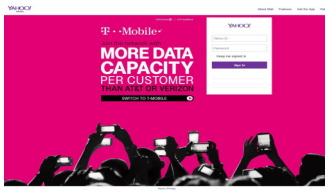


(e)

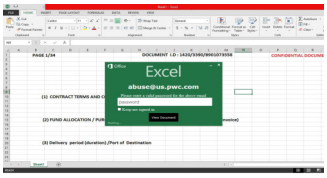


(f)

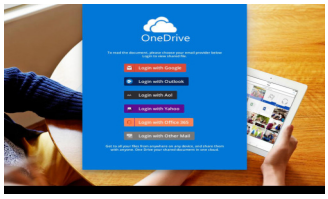
(g)



(h)

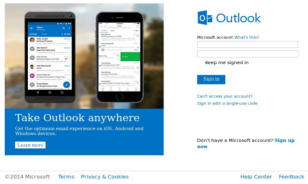


(i)



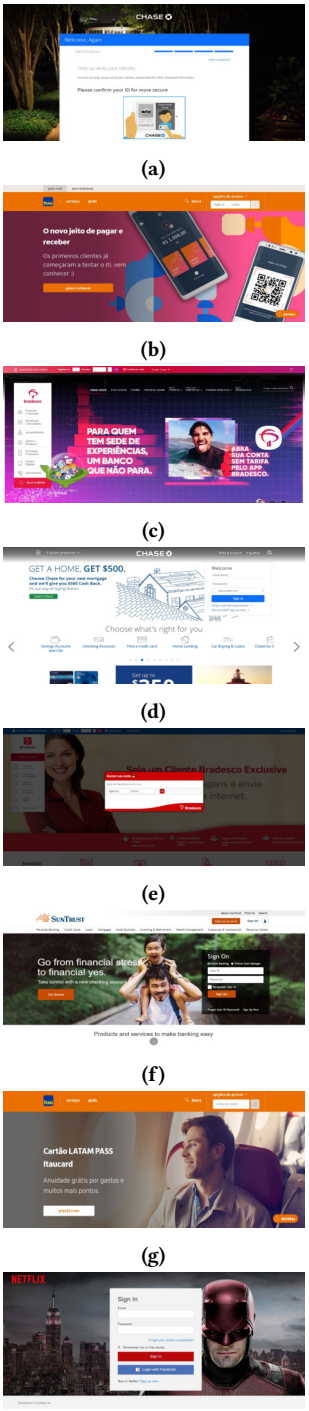
(j)

(k)



(l)

Figure 24: Examples of the phishing pages used in the online study where participants were asked if they think the appearances of these pages are trustworthy. The first column screenshots are the dissimilar examples. Only 3.02% of users (averaged on all screenshots) considered them trustworthy. The second column screenshots are the relatively more similar examples (with subtle differences) where 65.3% of users considered them trustworthy.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 25: Examples of the newly crawled phishing pages from PhishTank that are used to test zero-day pages. Additionally, all of these pages do not have counterparts in the targeted website's screenshots in the trusted-list and were not seen in training.