Inferring Phishing Intention via Webpage Appearance and Dynamics: A Deep Vision Based Approach

Ruofan Liu[1], Yun Lin[1]*, Xianglin Yang[1], Siang Hwee Ng[1], Dinil Mon Divakaran[2], Jin Song Dong[1]

School of Computing, National University of Singapore[1]

{dcsliny, dcslirf}@nus.edu.sg, {xianglin, sianghwee}@u.nus.edu, dcsdjs@nus.edu.sg

Trustwave[2]*; dinil.divakaran@trustwave.com*

Abstract

Explainable phishing detection approaches are usually based on references, i.e., they compare a suspicious webpage against a reference list of commonly targeted legitimate brands' webpages. If a webpage is detected as *similar* to any referenced website but their domains are not aligned, a phishing alert is raised with an explanation comprising its targeted brand. In comparison to other techniques, such explainable reference-based solutions are more robust to ever-changing phishing webpages. However, the webpage similarity is still measured by representations conveying only *partial* intentions (e.g., screenshot and logo), which (i) incurs considerable false positives and (ii) gives an adversary opportunities to compromise user confidence in the approaches.

In this work, we propose, PhishIntention, to extract precise phishing intention of a webpage by *visually* (i) extracting its brand intention and credential-taking intention, and (ii) interacting with the webpage to confirm the credential-taking intention. We design PhishIntention as a heterogeneous system of deep learning vision models, overcoming various technical challenges. The models "look at" and "interact with" the webpage for its intention, which are robust to potential HTML obfuscation. We compare PhishIntention with four state-of-the-art reference-based approaches on the largest phishing identification dataset consisting of 50K phishing and benign webpages. For similar level of recall, PhishIntention achieves significantly higher precision than the baselines. Moreover, we conduct a continuous field study on the Internet for two months to discover emerging phishing webpages. PhishIntention detects 1,942 new phishing webpages (1,368 not reported by VirusTotal). Comparing to the best baseline, PhishIntention generates 86.5% less false alerts (139 vs. 1,033 false positives) while detecting similar number of real phishing webpages. Our models and code are available at https://github.com/lindsey98/PhishIntention.git.

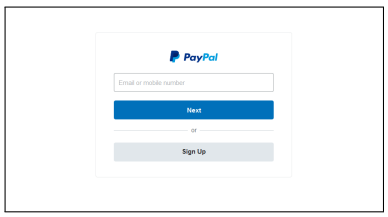
*Corresponding author

1 Introduction

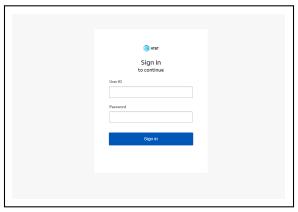
Phishing attack, one of the most common cyber attacks, causes huge financial losses every year [15]. The attacks and their defenses span across webpages [10,11,20,41,71], emails [29,30,38,59], and mobile applications [47,48]. Existing phishing detection techniques can be categorized into blacklist-based [8,51,52], classification-based [17,37,39,40,46,60,62,69], and referenced-based solutions [10,11,23,41,49,57,64].

Blacklist-based solutions such as Google Safe Browsing [2,8] and OpenPhish [6] use dynamic blacklists to track reported phishing URLs. In contrast, classification-based solutions [24,40,46,62,69], trained from a collected phishing dataset, perform binary predictions on phishing using the features extracted from a given webpage and its URL. Despite showing promising results in experiments, empirical study shows that they have limited performance on the ever-changing webpages in the wild [41]. The challenge is that the blacklists and collected phishing webpages used for training become obsolete very quickly [52]. Even worse, phishing kits keep evolving [21,27,53,54] and their creation is typically automated [21], resulting in a never-ending cat-and-mouse game between phishing attacks and defenses that are based on blacklists/classification.

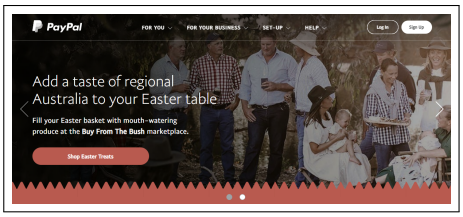
Reference-based phishing detection solutions are developed based on the intuition that, the key goal of a phishing attack is to deceive users with pages *visually* similar to legitimate websites. Thus, various reference-based solutions [10,11,16,23,41] are designed for detecting the *invariants* of phishing attacks, since a phishing webpage should be *semantically similar* to its targeted benign webpage. These approaches keep a list of references (e.g., screenshots, logos, etc.) representing well-known target websites (e.g., that of PayPal). Given a webpage $\mathcal{W}_p$, (i) if the representation of $\mathcal{W}_p$ is similar to that of a target webpage $\mathcal{W}_{tar}$, (ii) but the domain of $\mathcal{W}_p$ is different from the legitimate domain of $\mathcal{W}_{tar}$, $\mathcal{W}_p$ is reported as suspicious and its target brand provided as an explanation. For instance, if a webpage has a screenshot similar to that of the PayPal webpage, but its domain does



(a) PayPal login webpage.



(b) AT&T login webpage.



(c) A webpage disguising as PayPal.

Figure 1: Challenging examples for VisualPhishNet [10]: It matches Figure 1a and Figure 1b with high confidence despite they being different brands. In contrast, it cannot match Figure 1a and Figure 1c although they carry the same brand intention.

not align with PayPal's domain (e.g., `paypalverifysms.com` instead of `paypal.com`), a phishing alert is raised. Existing approaches use screenshots [10, 23] and logos [11, 16, 41] as the reference representations of target websites. However, they convey only partial intention of a phishing webpage, leading to false or missing alerts.

Screenshots as a reference do not perfectly capture brand intention, i.e., the intention indicating the company owning the webpage, since they contain many irrelevant details (e.g., pixel colors, embedded advertisements, etc.). Figure 1 shows a webpage wrongly reported by VisualPhishNet [10], which employs a deep learning model to compare the similarity of two screenshots. When VisualPhishNet finds a webpage's screenshot similar to that of a referenced webpage, it concludes that they convey the same webpage semantics. However, Figure 1a (reference page) and Figure 1b look similar, but convey different brand intentions. In contrast, Figure 1a and Figure 1c appear dissimilar, but convey the same brand intention (i.e., PayPal). Such visual challenges leads to both false positives and false negatives in VisualPhishNet (see Section 6).

Logos as a reference miss to capture credential-taking intention, i.e., the intention requiring a user to provide credentials. Figure 2 shows a false positive reported by Phishpedia [41], a deep-learning based approach which detects identity logo on a screenshot and compares it with logos in the reference list. In Figure 2, Phishpedia reports a benign webpage as phishing because it assumes that logos capture the full semantics of the webpage. As shown in Section 8, many webpage semantics can hardly be captured by logos alone. Benign webpages may refer to the big social media companies (e.g, Facebook) for login and registration via single sign-on method, article sharing, or advertisement promotion. We term such webpages as *misleading legitimacies*, which share plausibly similar brand intention to some target company. They pose a challenge for logo-based approaches. More importantly, an adversary can collect and disseminate misleading legitimacies to compromise user confidence in such solutions. Once users abandon the solution because of the false alerts, attackers can launch a new round of phishing campaigns.

Phishing pages are generally set up with credential-taking intention in various forms. Table 1 shows the general distribution and taxonomy of the 29,496 phishing webpages used

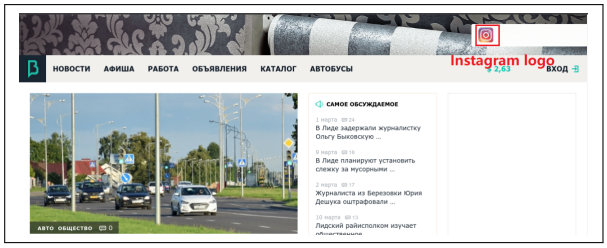


Figure 2: A false positive reported by Phishpedia; it reports the webpage as phishing Instagram, based on the logo on the top-right.

Table 1: Distribution and taxonomy of phishing webpages regarding the credential-taking intention. CRP stands for credential-requiring page.

# CRP phishing	# non-CRP phishing			Total
	Linked to CRP	Non-textual credential	False positive	
25403 (86.1%)	3,310 (11.2%)	704 (2.4%)	79 (0.3%)	29,496

in [41]. Overall, 86.1% of the webpages require credentials directly. For the remaining phishing webpages (i.e., 13.9%), 11.2% of the pages contain a link to a login/signup webpage, and 2.4% of the pages require sophisticated interactions such as scanning a QR code, solving a CAPTCHA, etc.

In this work, we propose PhishIntention, to precisely extract both brand and credential-taking intentions. Given a webpage, we design PhishIntention to infer (i) what brand information it represents, (ii) whether and how it requests user credential, and (iii) whether clicking certain button/menu on the screenshot can lure the users to provide their credentials. PhishIntention carries out both static and dynamic analyses to capture webpage intentions. Given a webpage, the static analysis extracts the brand and credential-taking intentions from the appearance of a webpage. Both intentions will be highlighted on the screenshot of a detected webpage (see Figure 10). The dynamic analysis further interacts with the webpage to verify whether it lures users to input credentials via its linked pages. Compared to state-of-the-art login form

detection techniques based on HTML analysis [22], PhishIntention uses computer vision technique to detect and interact with the webpage, which is more robust against HTML obfuscation (see experimental results in Section 8).

Technically, PhishIntention first extracts the abstract layout of a webpage screenshot consisting of all its salient UI components. Based on the layout, PhishIntention (i) selects the identity logo to identify its potential brand intention, and (ii) classifies the layout to recognize whether the webpage requires user credential, i.e., whether it is a CRP (credential-requiring page). PhishIntention predicts which UI component in a non-CRP may link to a CRP. By emulating user clicks on the webpage, PhishIntention retrieves and conducts further static and dynamic analysis on the new webpages.

We conduct extensive experiments to evaluate PhishIntention on a dataset of over 50K phishing and benign webpages. The experimental results show that PhishIntention significantly outperforms existing solutions such as VisualPhishNet [10], EMD [23], PhishZoo [11], and Phishpedia [41], in terms of detection capability and false-positive rate, by accurately recognizing brand and credential-taking intention of a webpage. Moreover, our experiments also demonstrate that PhishIntention is much more robust against misleading legitimacies and HTML obfuscation attack. Furthermore, we carry out a field study on the Internet for two months; PhishIntention discovered 1,942 new phishing webpages (including 1,368 not reported by VirusTotal). Comparing to the best baseline [41], PhishIntention generates 86.5% less false alerts (139 vs. 1,033 false positives) while detecting ∼6% less phishing webpages (1,942 vs. 2,071 true positives).

In summary, we make the following contributions:

- We propose a referenced-based phishing detection system that captures both brand intention and credential-taking intention. To the best of our knowledge, PhishIntention is the first work which analyzes both intentions in a systematic way for phishing detection.
- We address various technical challenges in detecting the intentions by orchestrating multiple deep learning models. By design, PhishIntention is robust against misleading legitimacies and HTML obfuscation attack.
- We conduct extensive experiments to evaluate PhishIntention. The experiments evaluate the overall and step-wise effectiveness, robustness against various adversarial attacks, and usefulness in practice.
- We implement PhishIntention with a phishing monitoring system. PhishIntention reports phishing webpages per day with the highest precision as compared to the state-of-the-art phishing detection solutions.

2 Threat Model

An attacker deploys an online phising webpage $\mathcal{W}_p$ disguising as the credential-requiring webpage (e.g., login or signup webpage) of a legitimate website $\mathcal{W}_{tar}$ (e.g., PayPal, Facebook, etc.). A user browsing $\mathcal{W}_p$ may be deceived to provide his/her credentials of website $\mathcal{W}_{tar}$ in $\mathcal{W}_p$. A solution needs to detect such phishing attacks while being robust against the following adversaries:

Misleading Legitimacy. Targeting reference-based solutions, an attacker can collect and disseminate legitimate webpages to users, causing the phishing detection system to generate false alerts. Such webpages can share plausible brand intention (e.g., having the same logo) with some well-known companies (e.g., Google and PayPal). By disseminating such *misleading legitimacies* to trigger false alerts, attackers can compromise user confidence in the solution, which lays the foundation for launching a new phishing campaign. Such misleading legitimacies are prevalent on the Internet, and our empirical results show that we can collect 3K misleading legitimacies within a week from the emerging new websites (see Section 9).

HTML Obfuscation. Targeting the solutions that use various HTML analysis techniques for detecting credential-taking intention, an attacker modifies the HTML code of the phishing webpages, while preserving the same appearance and dynamics to interact with the users.

Adversarial Attack against Deep Learning Models. Targeting any deep-learning based solutions, an attacker can generate adversarial samples to misguide the models to make wrong predictions. If successful, the phishing pages can evade a deep-learning based phishing detection solution.

PhishIntention mitigates the first adversary by significantly lowering the false-positive rate on misleading legitimacies; addresses the second by analyzing webpage screenshots, with the least dependence on HTML analysis; and addresses the third by adopting a model gradient-masking technique to nullify the popular gradient-based adversarial attacks.

3 Related Works

Classification-based and blacklist-based approaches. Blocking webpage visits using blacklists is a popular solution in the industry (e.g., using Google Safe Browsing [8] and OpenPhish [6]). The generation of a blacklist relies on a combination of automatic scanning and manual verification [14, 52]. However, generation of phishing websites is largely automated [21], and causes a time delay between blacklist updates and zero-day phishing emergence [52].

Many works are designed to craft features from URL, HTML code, screenshot, etc., to build a classifier for phishing detection. Rakesh et al. [62] construct a feature set including HTTP/HTTPS protocol type, URL length, etc., to distinguish phishing URLs from legitimate ones. Following their work,

Hung et al. [37] propose a deep learning sequential model called URLNet to classify URLs into phishing or benign. Cantina [69], Li et al. [40], and Lee et al. [39] aggregate both URL-based features and HTML-based features to achieve superior performance on their datasets. Readers may refer to a few surveys on classification-based solutions [34, 61, 63].

Reference-based approaches. Referenced-based solutions detect phishing attacks by referencing a set of representations (e.g., logo and screenshot) of well-known brands (Paypal, Amazon, etc.). Based on the representation forms, the solutions can be categorized into search-engine based, screenshot-based, and logo-based approaches.

Screenshot-based approaches explore different similarity measurements between two screenshots. Fu et al. [23] propose extracting a screenshot signature as reference. They adopt a dynamic programming algorithm, Earth Mover's Distance (EMD), to find the optimal match between two screenshots based on their color scheme. Following their work, Medvet et al. [49] and Rosiello et al. [57] compare visible text and visual similarity between HTML DOM tree and HTML tags in the screenshots. VisualPhishNet [10] is the latest screenshot-based reference solution, which trains a Siamese model to predict the similarity between two given screenshots.

Logo-based approaches address logo location and recognition problem. Afroz et al. [11] and Wang et al. [64] propose PhishZoo and VeriLogo, respectively. They employ SIFT algorithm [45] to detect and match the logo on a screenshot. Due to the limitation of SIFT algorithm, their approaches incur high false positive rate and large runtime overhead [41]. The most recent work Phishpedia [41] overcomes the challenge of recognizing logos by training two deep-learning based computer vision models, i.e., an object detection model and a Siamese model. Both PhishIntention and Phishpedia leverage computer vision models to detect phishing webpages, but they differ substantially in the following aspects:

Intention extraction. Phishpedia extracts only the brand intention of a webpage. Although it reports input boxes on the screenshot, they are not used to detect phishing pages. In contrast, PhishIntention extracts both brand intention and credential-taking intention, and further confirms the credential-taking intention via webpage interaction.

Technical contribution: Technically, Phishpedia uses only the logo to detect phishing. In contrast, PhishIntention orchestrates a system of deep learning models (detailed in the next two sections) to serve multiple purposes:

1) Inferring credential-taking intention by abstracting and predicting webpage layout;

2) Detecting whether a non-CRP can further lure a user to visit a CRP via both static and dynamic analyses;

3) Improving the logo detection technique with small number of samples (we achieve a better performance with ∼8K training screenshots, in comparison to Phishpedia trained with ∼29K screenshots) and the logo recognition technique with OCR-aided model architecture (see Q3 in Section 8 for details).

Algorithm 1: detect_phishing

Input : $url, \mathcal{S}, code, \mathcal{R}, t_{dep}, t_s$
Output : binary phishing result, target brand

```
if t_dep < 0 then
    return {False, Null}
// step 1: abstract webpage layout detection
awl = detect_layout(S)
// step 2: brand recognition
brand = match_logo(R, awl.logo, t_s)
if brand is null or url.domain ∈ brand.domains then
    return {False, Null}
else
    // step 3: classify CRP
    is_CRP = CRP_classify(S, awl)
    if is_CRP is true and url.domain ∉ brand.domains
      then
        return {True, brand}
    else
        // step 4: look for CRP (dynamic analysis)
        links = detect_potential_CRP(S, code)
        for link ∈ links do
            url_l, S_l, code_l = parse link
            // recursively continue the process
            {is_phishing, brand_tar} =
              detect_phishing(url_l, S_l, code_l, t_dep − 1)
            if is_phishing and
              url_l.domain ∉ brand_tar.domains then
                return {True, brand_tar}
return {False, Null}
```

4 PhishIntention: Design and Development

Figure 3 presents an overview of PhishIntention. Given a URL (along with its screenshot and HTML code), PhishIntention first extracts the Abstract Webpage Layout (AWL) for detecting both brand intention and credential-taking intention. For a webpage without credential-taking intention, PhishIntention interacts with the webpage to search for a link that in turn directs to a credential-requiring page (CRP).

Algorithm 1 (`detect_phishing`) lists down the steps. It takes as input a *url* (with its screenshot $\mathcal{S}$ and HTML code *code*), a list of protected brands $\mathcal{R}$, a CRP search depth t_{dep} for dynamic analysis, and a logo matching threshold t_s. It generates an output whether the *url* is a phishing webpage and the phishing target brand (if so). For each brand in $\mathcal{R}$, we maintain its logos and legitimate domains.

Step 1 (Section 4.1): Abstract Webpage Layout Detection. We extract a new form of webpage representation, i.e., Abstract Webpage Layout (AWL), describing the regions and po-

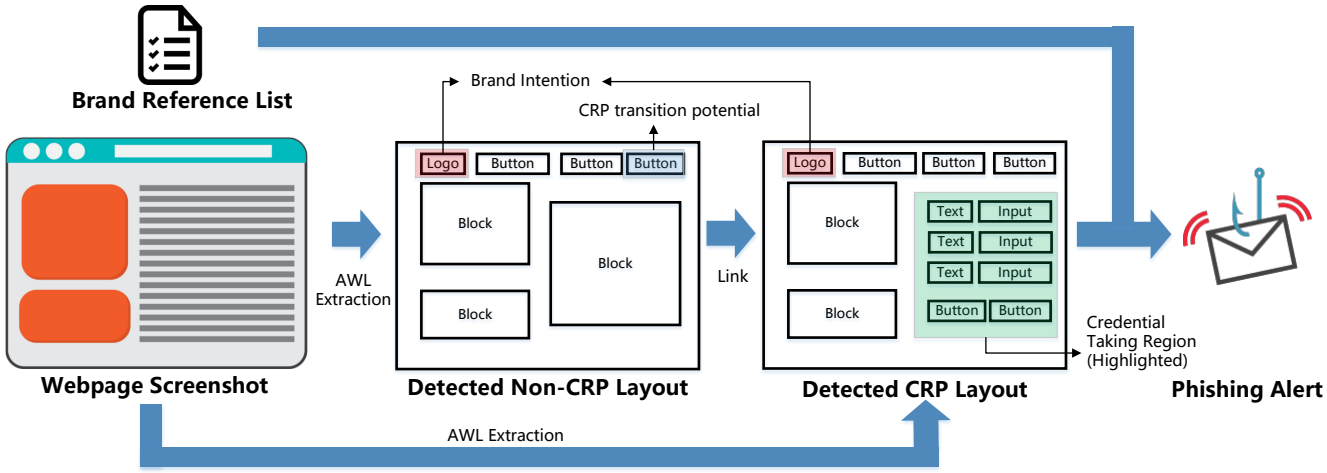


Figure 3: PhishIntention extracts screenshot layout to detect both its brand and credential-taking intentions. For a non-CRP, PhishIntention further infers the UI elements that could transition to a CRP. A phishing alert is generated when a webpage (1) intends to take credential, (2) conveys brand intention of a company r, while (3) its domain does not align with that of r.

sitions of salient UI components in the screenshot (as showed in Figure 6, and line 3 in Algorithm 1). The AWL helps in detecting logos and classifying whether a page requires credentials.

Step 2 (Section 4.2): Brand Recognition. We compare a logo in AWL (i.e., $awl.logo$, line 4 in Algorithm 1) and the logos of brands in $\mathcal{R}$. If there is no match with similarity greater than t_s, we report the webpage as benign. Otherwise, we further proceed to the next steps.

Step 3 (Section 4.3): CRP Classification. Subsequently, we build a CRP classifier that takes the screenshot and the AWL as input, and classifies whether the webpage requires user credentials. This step corresponds to line 8 in Algorithm 1. If the webpage is a CRP, but the domain of its URL does not align with any of the domains of the matched brand, we report it as a phishing page (lines 9-10).

Step 4 (Section 4.4): CRP-Transition Location. If the above mentioned conditions are not satisfied, we investigate whether any link/button on the webpage linked to a CRP, which can potentially lure users to provide their credentials. This step corresponds to lines 14-15 in Algorithm 1. We emulate user clicks on the reported links/buttons, and retrieve new redirected URLs along with their screenshots and HTML codes (line 14). Given a new URL (url_l), we further detect its phishing intention recursively (line 15). The recursive process continues until the predefined CRP search depth t_{dep} is exhausted (line 1-2 in Algorithm 1). We report the given URL as benign if all those links/buttons redirect to benign webpages.

In order to accomplish the above steps, we need to address the following technical challenges.

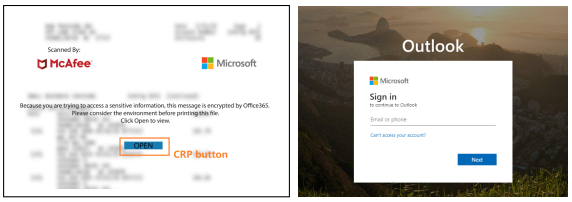
Challenge 1: Layout Definition and Extraction. Detecting the AWL of a webpage is a non-trivial task. *First*, visually salient components are hard to be fully formalized. Highlighting too fine-grained UI elements (e.g., detailed icon and text) to compose layout introduces irrelevant noises. In contrast, highlighting too coarse-grained UI elements (e.g., huge blocks) may miss salient intention-revealing information such as credential-taking UI components. *Second*, extracting a layout is technically challenging. Heuristics for visually salient HTML elements are hard to define. The HTML elements relevant to a page skeleton (e.g., *div* tag) may be visible or invisible given different Javascript and CSS frameworks.

(a) Bank of Scotland (b) A Russian Company

Figure 4: Example where Siamese model in Phishpedia mismatches two text-logos (from https://lyudyamolyudyah.ru/)

Challenge 2: Text-Logo Recognition. Logo recognition is basically a logo similarity measurement problem. It is a challenging task because (i) the logos under the same brand can be very diverse and (ii) the logos (especially text logo) under different brands can be visually similar. Lin et al. propose a Siamese training solution in Phishpedia [41] to address the first issue. However, the technique still suffers from the false positives caused by the similar text-logos under different brands. Figure 4 shows an example where the Siamese model proposed in Phishpedia mismatches the logo of Bank of Scotland with a Russian brand. They share similar appearances and color, but convey totally different brand intentions.

Challenge 3: Identifying and Confirming Credential-Taking Intention. An input box on a webpage does not necessarily take credentials, e.g., it could be a search box. More importantly, a phishing webpage portal may render no input for user credential, i.e., it may not be a CRP (see Figure 5). Instead, it presents a button to lure the user to input credential after clicking the button. For the latter issue (i.e., CRP-link on the webpage), the HTML code implementation can be quite



(a) Original screenshot (b) Forwarded CRP

Figure 5: A phishing webpage linking to a CRP. The CRP button on Figure 5a links to the webpage in Figure 5b.

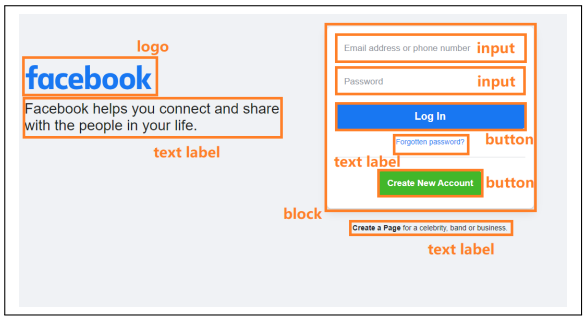


Figure 6: AWL of the Facebook login page

diverse; besides, attackers can evade using HTML obfuscation. These limit the effectiveness of HTML heuristics.

4.1 Abstract Webpage Layout (AWL)

Given a webpage screenshot $\mathcal{S}$, we define its *Abstract Webpage Layout*, $\mathcal{L}(\mathcal{S})$, as a set of *boxes*, representing visually salient rectangular regions on $\mathcal{S}$. Each box $\mathbf{b} \in \mathcal{L}(\mathcal{S})$ is described by a vector $\langle x, y, w, h, t \rangle$, where x, y represent its horizontal and vertical coordinates in $\mathcal{S}$, while w, h represent its width and height, and t is its UI type. In this work, we support five UI element types — identity logo, input box, button, text label, and block. Block indicates a salient region on the webpage screenshot which is none of logo, input box, button, and text label. The identity logo conveys the brand intention, whereas the spatial arrangement of those UI elements may further convey credential-taking intention. Figure 6 illustrates the AWL of a Facebook webpage. Each orange rectangle, with its type annotated, represents a visually salient region.

We design a data-driven approach to address the problem of ambiguous definitions of layouts. Specifically, we constructed a layout dataset of ∼9K webpage screenshots, annotated with regions and types of salient UI elements (Section 5.1). Subsequently, we train an object detection model to predict salient UI elements (and their types) on the screenshots. Technically, we regard each annotated component as an object on a screenshot (similar to a pedestrian or a car in a street view). We choose the Faster R-CNN model [56], which takes as input a screenshot, and generates as output a set of regions on the screenshot in the form of $\langle x, y, w, h, \mathbf{t} \rangle$, where, x, y, w, h represent the coordinate in horizontal position, vertical position, width, and height, respectively. And $\mathbf{t}$ is a one-hot vector where each dimension represents a specific classification type (e.g, logo, button, etc.). The deep learning model is trained to minimize the errors in type classification and region prediction on the training dataset.

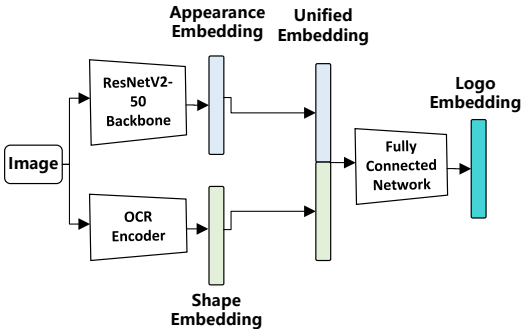


Figure 7: OCR-aided Siamese Model

4.2 OCR-aided Brand Recognition

An identity logo, if exists, is one of the components in the AWL. As mentioned in Section 4, a challenge in recognizing logo is that, it can be an icon-like image, a sequence of characters, or even both. To compare two logo images, Phishpedia's Siamese model [41] is not expressive enough to capture the fine-grained text shape (see Figure 4). In contrast, universally applying OCR (Optical Character Recognition) [32] cannot work on icon-like logos, and further incurs a high runtime overhead. Comparing to other deep learning solutions [56] which costs only a few milliseconds, an end-to-end deep OCR model [58] takes a prohibitive 0.5s on average for predicting the text of a single image.

We propose OCR-aided Siamese matching model to strike a balance in learning between appearance-based features and text-shape based features, which is also significantly more time-efficient than full OCR model. Our OCR-aided Siamese model is designed as depicted in Figure 7. The model has two branches: one branch is the ResNetV2-50 feature extractor from a pre-trained logo classification model which outputs appearance-based embedding $\mathbf{x}_{appear}$, the other branch goes through the encoder from a pre-trained OCR model which outputs text-shaped based embedding $\mathbf{x}_{shape}$. Here, the pre-trained OCR encoder captures text features over appearance-based features to recognize digits and characters. We take the ASTER encoder architecture [58] as our model.

The embeddings from the two branches are concatenated as a unified representation, i.e., $\mathbf{x}_{all} = [\mathbf{x}_{appear}; \mathbf{x}_{shape}]$. Finally, the overall embedding $\mathbf{x}_{all}$ is fed into a fully connected network, which is further transformed to the final logo embedding $\mathbf{x}_{logo}$. We jointly train all sub-modules through a logo classification task. During deployment, we compare two logos using the cosine similarity of their logo embeddings.

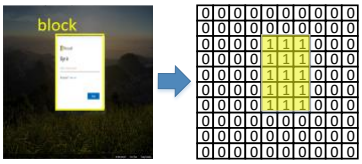


Figure 8: Pixel-matrix representation of block layer in the AWL input channel, assuming that the shape is 10×10.

Figure 9: The hybrid input for CRP classifier

4.3 CRP Classification

Defining HTML-based heuristics [22] (e.g., looking for specific HTML forms) can serve as a classifier for CRP. However, it is easy to be bypassed by HTML obfuscation attack. We design a vision-based CRP classifier which takes a screenshot and its AWL as a combined input, and is independent of HTML code implementation. The rationale is that the spatial arrangement of UI components (such as input boxes) may convey credential-taking intention (see Figure 6). Comparing to traditional image input which consists of 3-channel RGB image, we additionally construct M input channels (M is equal to the number of UI types) to encode the AWL. Let an RGB image I have shape of $N \times N$. Given a UI component $\mathbf{b}$ of type t (e.g., block), the region containing the element is specified by $< x, y, w, h >$. We assign the entries in pixel-matrix ($N \times N$) corresponding to the pixels overlapping with the region as ones and others as zeros. Figure 8 shows an example where we embed the type of block in a screenshot, which derives a channel of shape $N \times N$ and pixel entries assigned as 0s or 1s. Given M UI component types, we have M such channels. We stack those channels with the original RGB channels as the model input, as shown in Figure 9. Each additional channel corresponds to the layout of a type of UI component (e.g., button, logo, blocks, etc). The channels are fed into a model with ResNetV2-50 architecture [28] to classify whether a webpage requires user credentials.

To provide an explanation of the credential-taking intention, we highlight the credential-taking region on the screenshot, as shown in Figure 10. We also overlay a heatmap on the screenshot based on the model attribution technique, Score-CAM [66], which reasons out the contribution of each region

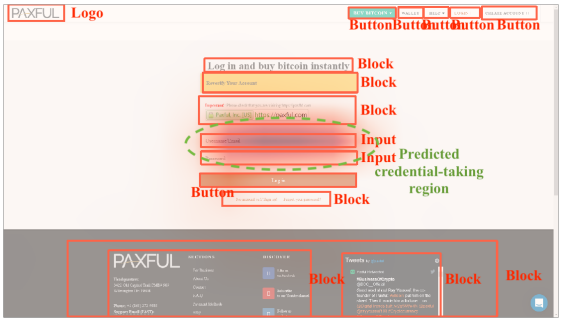


Figure 10: A screenshot annotated by PhishIntention: the highlighted region explains credential-taking intention

in the input to the output prediction. Technically, it measures the output sensitivity to perturbation on the model input. For more technical details, readers can refer to [66].

4.4 CRP-Transition Location

The landing phishing webpage may not always be a CRP. Sometimes, attackers provide a link or show a button to lure the victims to visit another CRP (see Table 1). We call such a link or button as a *CRP transition*.

To detect the CRP transitions on a webpage, we apply a hybrid solution, i.e., HTML heuristics as a pre-checking filter and a deep learning solution (independent of HTML) to predict the location of CRP transitions on the screenshot. Our heuristic reports the hyperlinks under either condition:
1) The English text is *exactly* "Login" and "Sign Up" (case-insensitive).
2) The non-English text translates to login and sign-up.

We search for the clickable DOM element with text that matches any of the keyword heuristics. If the pre-checking filter fails to locate any suspicious element, we resort to visual object detector.

We train an object detector (Faster R-CNN [56]) to predict the regions of CRP-transition based on screenshot, thereby being independent of HTML implementation. We construct a dataset consisting of screenshots annotated with CRP-buttons, i.e., UI elements linkable to CRP webpages (Section 5.1). Given that the number of the screenshots linkable to its CRP webpage is limited, we adopt a transfer-learning routine by pre-training an object detector on generated *pseudo* samples. Specifically, we search for icon/images of CRP-buttons such as buttons used for login and sign-up. Next, through the collected dataset, we identify the "hotspot regions" on the screenshots where CRP-buttons frequently appear. Then, we overlay the CRP-button images on webpage screenshots in the hotspot regions to produce a much larger *pre-training* dataset. After the pre-training, we fine-tune the object detection model based on original screenshots labelled with CRP-buttons.

The object detector allows us to assign a confidence score to each detected object. Hence, given a threshold for the maximum number of user clicks, K, our model recommends the top-K most likely UI components linkable to a CRP. We then emulate user clicks on those predicted components to visit and verify potential CRPs.

4.5 Defending Against Adversarial Attacks

To tackle adversarial attacks, we equip all the deep learning models with the gradient-masking technique proposed in [41]. Specifically, we change the ReLU activation function $f = max(0,x)$ to $f' = max(0, \alpha \cdot \lceil \frac{x}{\alpha} \rceil)$ to mask the gradients (where α is the discretization parameter).

5 Performance Evaluation

We conduct evaluations regarding the following research questions:

- **[RQ1] Phishing Detection Performance (Section 6):** How accurate and efficient is PhishIntention in detecting phishing webpages; and how does it perform in comparison to the baselines?
- **[RQ2] Credential-taking Intention Confirmation Performance (Section 7):** How accurately can PhishIntention locate CRP from a non-CRP webpage?
- **[RQ3] Model-wise Performance (Section 8):** How accurate is each deep learning model of PhishIntention?
- **[RQ4] Robustness Against Adversaries (Section 9):** Whether PhishIntention is robust against various adversaries:
  1) When attackers collect and disseminate misleading legitimacies, can PhishIntention well recognize them and preserve user confidence?
  2) When attackers build phishing webpages with HTML obfuscation, can PhishIntention still detect and confirm credential-taking intention?
  3) Whether attackers can successfully generate adversarial samples to compromise the deep learning models of PhishIntention?
- **[RQ5] Field Study (Section 10):** What is PhishIntention's performance in detecting phishing webpages in the wild?

We design one experiment for answering each of the above research questions. We refer to Appendix A.1 for details on hardware configurations for training of models. In the following sections, we first introduce the datasets used to train our deep learning models, then we describe each experiment with its settings, baselines, and experimental results. Given the complexity of experiment design (five sophisticated experiments sharing six datasets), we visually describe the connection among research questions, experiment designs, and their prepared datasets in Appendix A.2.

5.1 Training/Testing Dataset

The datasets for detecting AWL, recognizing logos, classifying CRPs, and locating CRP transitions are as follows.
For AWL Prediction and CRP Classification. We collected 9,010 webpage screenshots; the training:testing ratio as 9:1. We annotate each screenshot with two types of labels:

- **Layout Label:** Each screenshot is labelled with boxes describing various salient UI components (logo, button, text label, input, and block), to learn AWL. Overall, those 9,010 screenshots contain 9,540 logos, 64,386 buttons, 14,850 inputs, 7,383 labels, and 33,677 blocks.
- **CRP Label:** Each screenshot is labelled to indicate whether it requires user credential or not.

For OCR-aided Siamese. We use Logo2K+ dataset [67] to pre-train our OCR-aided Siamese model, containing 167,140 logo samples across 2,341 brands. In addition, we use the Synth90k [33] and SynthText [26] datasets to pre-train the OCR branch. We select 1,000 screenshots with logos from the phishing webpage dataset, 1,000 from legitimate websites, and crop their logos as testing dataset.
For CRP-Transition Location. We overlay 63 CRP-button images on to 10,000 screenshots; this is done only for pre-training. After pre-training, we use 4,843 non-CRP screenshots collected from real-world non-CRP phishing webpages and the homepages of well-known websites (see Appendix A.2) to train deep CRP locator. 3,310 non-CRP phishing and 1,003 non-CRP legitimate are used for testing.

5.2 Reference List & Click Emulation

We prepare a target list that includes 3,061 logos from 277 well-known brands (also used in [10,41]). The brands cover a wide range of sectors including banking, online shopping, social media, etc. We emulate user clicks via Helium library [9] to interact with online webpages. Helium is a tool built upon Selenium which has high-level API design. We use its eager mode to reduce webpage loading time.

6 RQ1: Phishing Detection Experiment

6.1 Settings

To evaluate PhishIntention's performance in detecting brand intention and credential-taking intention, we use 25,403 CRP phishing webpages (see Table 1) from the phishing webpage dataset. We randomly sample an equal number of webpages

Table 2: Baseline description (see Section 3 for details)

Solution	Reference Representation	Description
EMD [23] (TDSC'06)	Screenshot	Uses EMD algorithm to compute the similarity between two screenshots' colour distributions.
VisualPhishNet [10] (CCS'20)	Screenshot	Trains a deep Siamese model to report the similarity score of two screenshots.
PhishZoo [11] (ICSC'11)	Logo	Uses SIFT algorithm to report whether a screenshot contains a given logo.
Phishpedia [41] (USENIX Sec'21)	Logo	Employs an object detection model to extract logos from screenshots and a Siamese model to compare the similarity between two logos.

from the benign webpage dataset. We evaluate the precision/recall of phishing detection.

Note, PhishIntention searches for a CRP from a non-CRP. Thus, it is not possible to evaluate the overall pipeline on traditional dataset, as many collected phishing webpages are dead. We discuss PhishIntention's performance of dynamic analysis in Section 7.

6.2 Baselines

Table 2 lists four reference-based existing techniques considered in this study. We select two screenshot-based solutions and two logo-based solutions. EMD and PhishZoo are well-known early reference-based approaches. VisualPhishNet and Phishpedia are the latest state-of-the-art screenshot-based and logo-based approaches, respectively.

Reference selection. We construct a target list of 277 brands for analysing all the five approaches. We equip the screenshot-based approaches EMD and VisualPhishNet with 9,334 referenced screenshots, and the logo-based approaches PhishZoo and Phishpedia with 3,061 referenced logos (see Section 5.2).

Implementation. We use the implementation of EMD and PhishZoo published in [3]. We implement Phishpedia based on [41]. For VisualPhishNet, we use the implementation made publicly available [5].

6.3 Results

Figure 11 plots the ROCs in for the five solutions. The logarithm scale helps to observe their performances (in terms of recall or true positive rate) at the practical requirement of low false positive rates (FPR), e.g., at FPR of 10^{-3}. In comparison to the best performing baseline solution Phishpedia, PhishIntention achieves a higher accuracy on overall benign/phishing webpage dataset. At a low FPR of 10^{-3}, PhishIntention achieves a recall of 0.9, which is *more than twice the recall of Phishpedia*. To put it differently, at the recall of 0.9, PhishIntention incurs a FPR that is two orders of magnitudes less than that of Phishpedia ($\sim 10^{-3}$ versus $\sim 10^{-1}$). The remaining solutions have negligible recall values at an FPR of 10^{-3}.

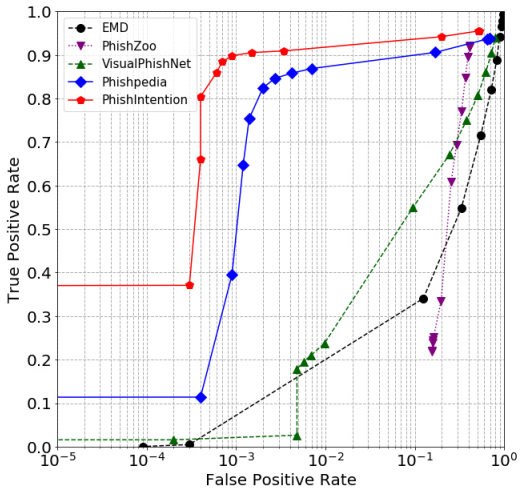


Figure 11: ROCs of EMD, PhishZoo, VisualPhishNet, Phishpedia, and PhishIntention

(a) Matched logo (Visa)

(b) Logo from NSK

Figure 12: Phishpedia's Siamese model wrongly matching NSK logo (from nsk.cyrek.xyz) to Visa logo

It shows that PhishIntention improves the reference-based phishing detection capability beyond the existing solutions. Lastly, the mean/median runtime overhead of PhishIntention is relatively low — PhishIntention incurs 0.58s/0.69s, whereas Phishpedia incurs 0.39s/0.41s, EMD 1.23s/1.20s, PhishZoo 65.45s/21.93s, and VisualPhishNet 0.26s/0.23s.

6.4 Qualitative Analysis

We now qualitatively compare PhishIntention with Phishpedia and VisualPhishNet to further understand how our approach outperforms the baselines. More examples can be found in [7].

Advantages of PhishIntention over Phishpedia. We observe that PhishIntention outperforms Phishpedia for three reasons: (1) accurate credential-taking intention extraction (for higher precision), (2) more accurate OCR-aided logo matching (for higher precision), and (3) more accurate logo detection (for higher recall). As for the first reason, we have shown in Figure 2 that only detecting brand intention is insufficient, we discuss the other two reasons here.

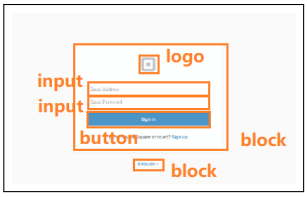
The extra OCR branch helps PhishIntention to reduce false

positives on text-based logos made by Phishpedia. Figure 12 shows a false positive reported by Phishpedia which matches the logo of NSK with that of Visa. In contrast, PhishIntention distinguishes text-based features well.

(a) Original screenshot

(b) Matched logo (Square)

(c) Phishpedia Results



(d) PhishIntention Results

Figure 13: A phishing webpage with color of logo similar to that of background (https://squareup.com/login)

Furthermore, we observe that PhishIntention is more sensitive than Phishpedia in recognizing brand logos because PhishIntention is trained on more diverse UI components (i.e., logo, button, input, label, and block). Figure 13 shows a phishing example whose target brand is Square (https://squareup.com/login). Figure 13a shows the screenshot of the phishing webpage and Figure 13b shows the matched logo by PhishIntention. Figure 13c and Figure 13d depict the results from Phishpedia and PhishIntention, respectively. We observe that PhishIntention recognizes complete salient UI components such as logo, input, button, and block, while Phishpedia reports neither logo nor input, despite the fact that the object detection model of Phishpedia is trained using a ∼29K labelled screenshot dataset; PhishIntention, on the other hand, is trained on only ∼8K training dataset.

Advantages of PhishIntention over VisualPhishNet. PhishIntention outperforms VisualPhishNet because the similarity between screenshots can be confusing when (1) the test webpage conveys the same brand intention yet having a different appearance from template, or (2) the webpage conveys a different brand intention but have a similar appearance as template (as shown in Figure 1). We provide more of such examples in [7].

7 RQ2: CRP Location Experiment

7.1 Settings

We evaluate PhishIntention's CRP locating technique with two datasets (see Section 5.1): (1) the 3,310 non-CRP phishing webpages containing a CRP button, and (2) 1,003 legitimate non-CRPs, each of which has a ground-truth URL redirected to a CRP page. For the example of the latter, under the domain of paypal.com, we pick a non-CRP such as `https://www.paypal.com/home`, and attach it with a ground-truth URL of CRP (e.g., `https://www.paypal.com/signin`). We use *completeness* as the performance measurement. Specifically, let the number of webpages reported with true CRP button (or true CRP URL) be M and the total number of webpages in the experiment be N; *completeness* is defined as $\frac{M}{N}$. For the legitimate webpages, we evaluate how complete PhishIntention is in discovering the ground-truth CRP URLs. Note that, the non-CRP phishing webpages are inaccessible. Thus, we evaluate how complete can PhishIntention report the ground-truth CRP-transition on the screenshot.

We run CRP locator on the above two datasets in three modes, i.e., (1) only with HTML heuristics, (2) only with deep learning model (i.e., Faster R-CNN model), and (3) with both HTML heuristics and deep learning model.

7.2 Results

Table 3 gives the result. The HTML heuristics and the deep learning model complement with each other. The combined approach of HTML heuristics and deep learning model has the best performance to locate CRPs on the non-CRPs. For the non-CRP phishing dataset, deep learning model alone achieves an acceptable completeness of 83.1%. Note that, non-CRP phishing webpages are dead so we cannot confirm their CRP links. Therefore, we can only apply the deep learning solution on their screenshot.

Table 3: Performance of dynamic analysis to locate a CRP from a non-CRP

Approach	Dataset	Completeness
HTML only	Legitimate	83.2%
	non-CRP Phishing	/
Deep learning only	Legitimate	72.9%
	non-CRP Phishing	**83.1%**
HTML+Deep learning	Legitimate	**93.3%**
	non-CRP Phishing	/

We also investigate the samples where PhishIntention does not work well. We observe that the webpages with uncommon login-keyword or login-icon can mislead the CRP locator. Figure 14 presents a screenshot of a legitimate website. PhishIntention predicts two boxes as its CRP buttons (boxes in orange). However, one button is to change language and the other is to search inside the website. The real CRP button is with the name of "IdCorreios", which uses an uncommon icon to represent the login semantics. Moreover, its HTML code does not conform to our predefined credential-requiring keywords in HTML heuristics. A remedy is to fine-tune our model based on the informative samples. We will investigate this in our future work.

Figure 14: PhishIntention fails to locate CRP transition

8 RQ3: Evaluating Model-wise Contributions

We evaluate the following questions in this section:

- **Q1:** What is the prediction accuracy of each individual model, i.e., AWL detection model, CRP transition location model, CRP classifier, and the OCR-aided Siamese model?
- **Q2:** How is the performance of PhishIntention in detecting logo comparing to that of Phishpedia?
- **Q3:** How effective is PhishIntention's OCR-aided Siamese model in comparison to Phishpedia's Siamese model?
- **Q4:** How effective is PhishIntention's hybrid input based CRP classifier in comparison to individual input classifier?
- **Q5:** How effective is PhishIntention's CRP classifier in comparison to naive input-box information?

Settings. For the above models, we use the datasets mentioned in Section 5.1. We use mAP (mean Average Precision), a measurement widely used in computer vision community, to evaluate object detection models i.e. layout detection and CRP button/link detection. For CRP classifier, we evaluate its classification accuracy.

Q1) Individual Model Accuracy. Table 4 shows the accuracy of all models. Overall, each individual model achieves good results. The AWL detector achieves mAP of 56.7 averaged over all UI categories and mAP of 59.5 on logo category. CRP-Transition Locator achieves a high mAP of 44.6.

Q2) Logo detection performance. We compare the logo detection capabilities of PhishIntention and Phishpedia in the first two rows of Table 4 (second column). PhishIntention achieves significantly higher performance (by about 13 mAP points); this is because the model in PhishIntention is trained with more diverse types of UI elements (than in Phishpedia).

Q3) Effectiveness of OCR-aided Siamese Model. We compare OCR-aided Siamese model and the Siamese model used in Phishpedia in the last two rows of Table 4; the OCR-aided Siamese model matches ∼5.5% more logos.

Table 4: The testing performance of each deep learning model. The mAP measurement is under the IoU thresholds 0.5:0.95.

Model	Overall mAP	Logo mAP	Prediction Acc	Matching Acc
Layout Detection	**56.7**	**59.5**	/	/
Logo/Input Detection (in Phishpedia)	54.7	46.6	/	/
CRP-Transition Locator	**44.6**	/	/	/
CRP Classifier	/	/	**0.950**	/
OCR-Aided Siamese	/	/	/	**0.891**
Traditional Siamese (in Phishpedia)	/	/	/	0.835

Table 5: Comparing CRP classifier with different inputs

CRP Classifier Type	Train Accuracy	Test Accuracy
Layout-only classifier	94.8%	90.0%
Screenshot-only classifier	**100.0%**	89.0%
Combined classifier	99.3%	**95.0%**
Input box heuristics	/	60.8%

Q4) Effectiveness of Hybrid Input. We train CRP classifiers with layout-only as input and screenshot-only as input, and compare them with the CRP classifier with hybrid inputs. Table 5 shows the results. Typically, training only on screenshot leads to over-fitting because of learning detailed pixel-level features. In contrast, training on more general features, i.e. layout, allows the model to generalize better but raises challenges in model fitting. Combining both allows us to strike a balance and achieve the best performance.

Q5) Effectiveness of CRP Classifier. We evaluate whether using input-boxes on screenshots is sufficient to predict credential-taking intention. Here, we use Phishpedia as a baseline; it can report both logo and input boxes, and we use the input box prediction as an indicator to predict CRP. The last row in Table 5 gives the result. Such a simple heuristic can only achieve classification accuracy of 60.8%, performing much lower than the above three CRP classifiers.

9 RQ4: Robustness Against Adversaries

We evaluate the robustness of PhishIntention against adversaries of misleading legitimacy, HTML obfuscation, and adversarial attack on deep learning models.

9.1 Settings

i) Misleading Legitimacy. We collect and verify 3,049 misleading legitimate websites (from Apr 9, 2021 to Apr 16, 2021) from the emerging new websites reported by CertStream [1] based on any of the three conditions: 1) media/news/blog websites with links to Facebook, Twitter, Youtube, etc., 2) websites allowing sign-in or registration via Facebook, Google, etc., and, 3) websites with advertise-

ment of companies such as Amazon, Microsoft, etc. We run PhishIntention and four baselines (Section 6) to evaluate their robustness against misleading legitimacies.

ii) HTML Obfuscation Adversary. PhishIntention analyzes webpages for 1) CRP detection (Section 4.3) and 2) CRP-Transition Location (Section 4.4). Our analysis consists of light-weight HTML analysis and computer vision model. In this experiment, we compare PhishIntention with XDriver [22], the latest HTML-based technique to discover and detect login/signup form, based on their robustness against the following HTML obfuscation techniques:

- **HTML Form Obfuscation (for CRP classification):** We apply squatting techniques [19,60] on HTML forms, e.g., by replacing "login" and "sign up" with "l0gin" and "sigh up".
- **URL Obfuscation (for CRP location):** We randomly rename the URLs linking to the CRPs while preserving the CRP linkability.

We obfuscate the webpages used in CRP location experiments (Section 7) and CRP classification experiments (Section 7), and evaluate the model accuracy after the obfuscation. The obfuscations are designed to nullify all the state-of-the-art HTML heuristics (including ours), which are *practical* solutions (1) taking the minimal efforts from the attackers and (2) preserving the website appearance.

iii) Deep Model Adversary. We have two classifier-like models (CRP classifier and Siamese model) and two object detectors (layout detection model and CRP location model). We choose five popular adversarial attacks on the classifiers and DAG [70] attack on the object detectors. The five popular adversarial attacks are I-FGSM [25,36], I-StepLL [36], C&W L2 [18], DeepFool [50], and BPDA [13]. While I-FGSM, I-StepLL, C&W L2, and DeepFool depend on model gradient to conduct attack, BPDA [13] is a gradient-recovering technique to recover masked gradients. DAG [70] is an adversarial attack designed for object detection model, and its attack is also based on model's gradients.

We evaluate the robustness of each of PhishIntention's models against these attacks.

9.2 Results

i) Misleading Legitimacy Adversary. Figure 15 shows the performances in terms of phishing detection rate on the phishing/bengin webpage dataset (see Section 6) and the FPR on the misleading legitimate dataset. Observe that, while the baselines suffer from high FPRs (45.5% to 60.1%, see Figure 15), the false-positive rate of PhishIntention is significantly much lower, i.e. 5.10%.

ii) HTML Obfuscation Adversary. Table 6 presents accuracy of CRP classification and CRP location before and after HTML obfuscation. The accuracy of Xdriver [22] drops much more significantly as the technique depends purely on HTML

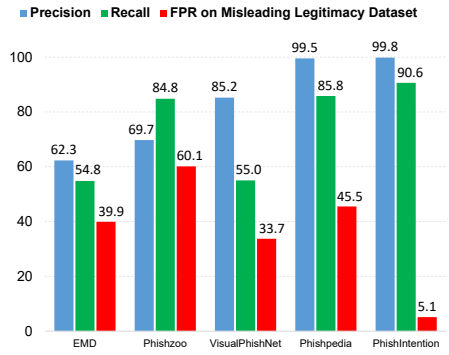


Figure 15: Best precision and recall on traditional webpage dataset, and false positive rate on misleading dataset.

Table 6: Robustness Against HTML Obfuscation

Solution	Classification Accuracy		Location Accuracy	
	Before	After	Before	After
PhishIntention	95.0%	95.0%	93.3%	72.6%
Xdriver	75.4%	59.2%	39.3%	3.4%

code. In contrast, PhishIntention largely preserves its classification and location accuracy even when its HTML heuristics are bypassed, thanks to the computer vision solution.

iii) Deep Model Adversary. Our experimental results show that all models of PhishIntention are robust against I-FGSM, I-StepLL, C&W L2, DeepFool, and DAG. Table 7 and Table 8 shows that the accuracy loss under attacks is much lower after the defense.

10 RQ5: Phishing Discovery Experiment

10.1 Runtime Configurations

We set the similarity threshold t_s to 0.87 based on F-score evaluated on experimental dataset (See Table 9), and let the threshold t_{dep} in Algorithm 1 be 1; i.e., we visit only one-level adjacent webpages. We set the timeout to load a new webpage to 2 seconds, and the threshold for the maximum number of times to emulate user clicks K to 3. We limit the interaction depth and time for efficiency and ethical considerations (by avoiding exhaustive testing on benign webpages).

10.1.1 CertStream URLs

We choose CertStream [1] for feeding the webpage crawling system. CertStream is a free online service which provides users with real-time feeds of URLs issued with new certificates. We then feed those URLs (along with their screenshot and HTML code) to various phishing solutions and analyze (1) how many phishing webpages are detected? and (2) How precise is each solution? The real-world phishing experiment complements offline experimental dataset as phishing kits

Table 7: Defense effectiveness on adversarial attacks on classifiers: The accuracy loss with and without the defense (N.D. for no defense and Def for defense). The original accuracy of CRP classifier and OCR Siamese model are 0.95 and 0.98.

Model	I-FGSM		I-StepLL		C&W L2		DeepFool		BPDA	
	N.D.	Def	N.D.	Def	N.D.	Def	N.D.	Def	N.D.	Def
CRP Classifier	↓0.95	↓0.0	↓0.95	↓0.0	↓0.40	↓0.0	↓0.95	↓0.0	↓0.95	↓0.07
OCR Siamese Model	↓0.98	↓0.0	↓0.98	↓0.0	↓0.98	↓0.0	↓0.98	↓0.0	↓0.98	↓0.01

Table 8: Defense effectiveness on adversarial attacks on object detectors

Model	Original mAP	DAG without Defense	DAG with Defense
AWL Detector	56.7	↓8.6	↓0.4
CRP-Transition Locator	44.6	↓13.4	↓0.6

keep evolving, which allows us to evaluate approaches on the most recent phishing attacks.

10.1.2 Baselines and Manual Validation

We run the baselines (listed in Table 2) with the configurations described in Section 10.1. The discovery experiment is carried out for two months starting from April 2021. We hired two students with two years of web security experience to manually evaluate the reported phishing URLs. Specifically, we built a Telegram service to push the detected phishing webpages (screenshots with annotated target brands) to their mobiles; and the students provided feedback independently on each sample, reporting phishing, benign, or unsure. The students see only screenshot annotated with explanation; we also provided them virtual machine to visit corresponding URL when interactions were needed. For the inconclusive samples, we invited another security expert to discuss with them to come to a final consensus. Furthermore, for each confirmed phishing webpage, we use VirusTotal [4] service to investigate whether any one of its 89 engines (e.g., Google Safe Browsing) report the given webpage as malicious or phishing. If not, we report it as a phishing attempt *not reported by VirusTotal*.

10.1.3 System Design

Figure 16 presents the distributed system we design for this experiment. Our crawler keeps crawling the URLs fed from CertStream as a "producer". Moreover, we use multiple VPNs to access the same URL to mitigate the effect of cloaking techniques [31, 72]. The URL, its HTML code and its screenshot are stored in a database node. Each phishing solution is deployed in a separate node as a "consumer" to predict on the webpages. The rationale is that the speed of crawler is much faster than some phishing solutions. Once a phishing solution reports a phishing webpage, it will push the notification through Telegram to the mobiles of our evaluators. Their labels on the webpages will be sent back to the server.

Table 9: Threshold t_s Selection

Threshold	0.7	0.83	0.85	0.87	0.90	0.93
F-score	0.8278	0.9973	0.9978	**0.9979**	0.9977	0.9971

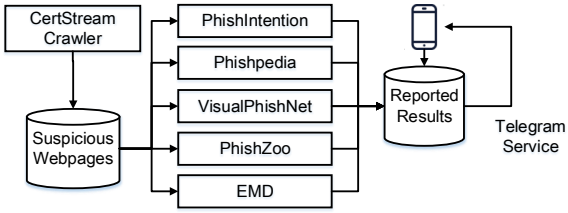


Figure 16: Distributed system designed for the field study.

10.2 Results

Figure 17 presents the results of our phishing discovery experiment. Evidently, PhishIntention and Phishpedia outperform the rest in reporting real phishing webpages; whereas, the number of false positives of other solutions are way too high for them to be of practical use. Specifically, Phishpedia reports 3,104 phishing webpages, of which 2,071 are true positives, and 1,514 are not reported by VirusTotal. In contrast, PhishIntention reports 2,081 as phishing, with 1,942 of them being true positives and 1,368 not reported by VirusTotal. It indicates that many emerging phishing webpages are missed by all the industrial anti-phishing solutions in VirusTotal (e.g., Google Safe Browsing and Kapaskey), confirming the results in recent studies [52, 55].

PhishIntention has a much higher precision than Phishpedia (6.7% vs. 33.3% FPR). In other words, PhishIntention generates 86.5% less false alerts than Phishpedia. Among the 1,942 reported real phishing webpages by PhishIntention, 138 of them (i.e., 7.1%) are contributed by dynamic analysis. Figure 18 shows the number of targeted brands of all the discovered phishing webpages detected by PhishIntention. These phishing webpages are distributed among 136 brands. The distribution has a long-tail shape, wherein the top 10 brands cover 70.1% phishing webpages and top 20 brands cover 80.4% phishing webpages.

Moreover, our investigation shows that Phishpedia and PhishIntention have some overlapping as well as different results; this is depicted in Figure 19. We observe that, to achieve much higher precision, the trade-off that PhishIntention makes is considerably low — it misses only 6.22% of phishing webpages detected by Phishpedia.

The reasons for PhishIntention's false positives and false negatives are covered in our discussion in Section 6.4 and our website [7]. However, we observe that PhishIntention has additional false negatives in the field study. Unlike Phishpedia,

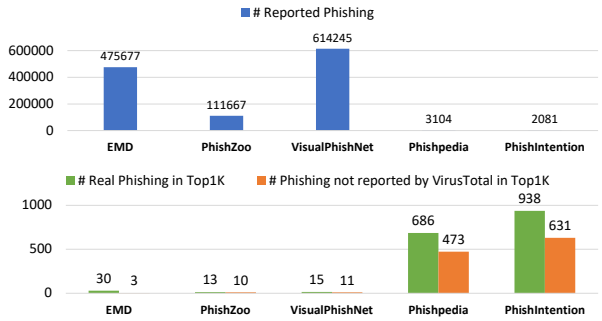


Figure 17: Phishing discovery results.Top-1K webpages are selected based on confidence provided by each solution.

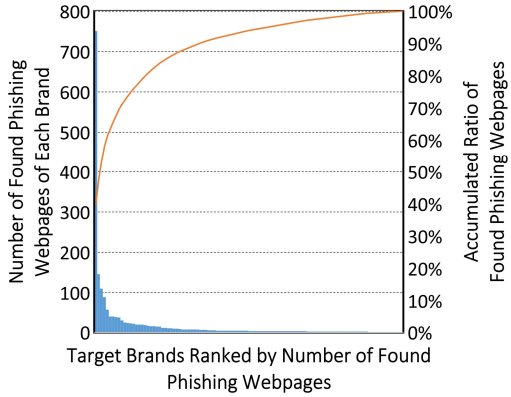


Figure 18: Distribution of brands of discovered phishing webpages. Top five brands are Microsoft (751), Facebook (146), HSBC Bank (110), Amazon (89), and Instagram (58).

PhishIntention will not report dead phishing webpages, as it needs to interact with non-CRPs to confirm the credential-taking intention. However, the system design (Figure 16) may sometimes prevent PhishIntention from interacting with a phishing webpage in time before it expires. As per Figure 16, to ensure all the approaches work on the same webpages as much as possible, our crawling system (as producer) stores downloaded webpages in a database, and different phishing solutions (as consumers) analyze the stored webpages. This causes some webpages to wait in the queue for hours (potentially making a few of them obsolete) before being processed by the phishing solutions. Note that, this design only affects PhishIntention as other solutions work on the offline webpage screenshots. PhishIntention may realize less false negatives if allowed to interact with webpages in real-time.

11 Discussions

Diversified Intention. The intention of a webpage to require credential can be more diversified than those considered in this work. For example, some phishing webpages can use a new form of UI, (e.g., QR code), to retrieve credentials. Thus, a more enriched intention detector is required in the future.
Diversified Interaction Model. Our approach requires confirming the credential-requiring intention of a webpage through webpage interaction. However, some phishing webpages require different interaction methods to proceed to the next page, e.g., a CAPTCHA can block the interaction of PhishIntention. A remedy is to build a more sophisticated *webpage interaction model*.
Practical Deployment. The effort to train PhishIntention can be referred in Appendix A.1. To maintain the model performance (e.g., to learn new logos/layouts), we can employ online learning techniques [12] on the models at a fixed frequency (e.g., every three months) with the collected false positives and false negatives.
Extension to Other Web-security Problems. We envision PhishIntention's framework to be useful in other applications:

- Drive-by-download detection, to recognize malicious intention of the downloading-buttons.
- Malicious link detection, to recognize the redirection links to other illegal webpages.

By interacting with those webpages, an agent can help us confirm their intentions by actually playing as a victim in the sandbox, improving the confidence and the explainability.

(a) True Positives (b) False Positives

Figure 19: Overlaps and differences in phishing webpages reported by Phishpedia (red) and PhishIntention (blue).

12 Conclusions

We proposed PhishIntention to identify and explain phishing webpages through static and dynamic analysis. The heterogeneous solution extracts brand and credential-taking intentions of phishing webpages. Our experiments on a large dataset and field study demonstrated the effectiveness of PhishIntention over existing solutions. In the future, we will apply PhishIntention into an online phishing monitoring system to collect active phishing kits, and study their runtime behaviors with various program analysis techniques [42–44, 65].

Acknowledgement

We thank the anonymous reviewers who help us to improve this work. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd, the Minister of Education, Singapore (T2EP20120-0019,

T1-251RES1901, MOET32020-0004), the National Research Foundation Singapore through its National Satellite of Excellence in Trustworthy Software Systems office (NSOE-TSS2019-05).

References

[1] CertStream Service. https://certstream.calidog.io/.

[2] Google Safe Browsing API. https://developers.google.com/safe-browsing/v4.

[3] Phishpedia Site. https://sites.google.com/view/phishpedia-site/home.

[4] VirusTotal. https://www.virustotal.com/gui/.

[5] VisualPhishNet Github Repository. https://github.com/S-Abdelnabi/VisualPhishNet.

[6] OpenPhish. https://www.openphish.com/, October 2020.

[7] Anonymous PhishIntention Website. https://sites.google.com/view/phishintention/home, 2021.

[8] Google Safe Browsing. https://safebrowsing.google.com, 2021.

[9] Helium. https://github.com/mherrmann/selenium-python-hel, 2021.

[10] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity. In *Proc. ACM CCS*, 2020.

[11] Sadia Afroz and Rachel Greenstadt. Phishzoo: Detecting phishing websites by looking at them. In *2011 IEEE fifth International Conf. on Semantic Computing*.

[12] Terry Anderson. *The theory and practice of online learning*. Athabasca University Press, 2008.

[13] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, pages 274–283. PMLR, 2018.

[14] Simon Bell and Peter Komisarczuk. An Analysis of Phishing Blacklists: Google Safe Browsing, OpenPhish, and PhishTank. In *Proc. Australasian Computer Science Week Multiconference*, 2020.

[15] Marzieh Bitaab, Haehyun Cho, Adam Oest, Penghui Zhang, Zhibo Sun, Rana Pourmohamad, Doowon Kim, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, et al. Scam pandemic: How attackers exploit public fear through phishing. In *2020 APWG eCrime*, pages 1–10. IEEE.

[16] Ahmet Selman Bozkir and Murat Aydos. Logosense: A companion hog based logo detection scheme for phishing web page and e-mail brand recognition. *Computers & Security*, 95:101855, 2020.

[17] Ebubekir Buber, Banu Diri, and Ozgur Koray Sahingoz. Nlp based phishing attack detection from urls. In *International Conference on Intelligent Systems Design and Applications*, pages 608–618. Springer, 2017.

[18] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE S&P*, pages 39–57.

[19] Sen Chen, Lingling Fan, Chunyang Chen, Minhui Xue, Yang Liu, and Lihua Xu. Gui-squatting attack: Automated generation of android phishing apps. *IEEE Trans. on Dependable and Secure Computing*, 2019.

[20] Igino Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli. Deltaphish: Detecting phishing webpages in compromised websites. In *Proc. ESORICS*, pages 370–388, 2017.

[21] Marco Cova, Christopher Kruegel, and Giovanni Vigna. There is no free phish: An analysis of "free" and live phishing kits. *WOOT*, 8:1–8, 2008.

[22] Kostas Drakonakis, Sotiris Ioannidis, and Jason Polakis. The cookie hunter: Automated black-box auditing for web authentication and authorization flaws. In *Proc. ACM CCS*, pages 1953–1970, 2020.

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

[24] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. A framework for detection and measurement of phishing attacks. In *Proc. ACM workshop on Recurring malcode*, pages 1–8, 2007.

[25] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[26] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proc. IEEE CVPR*, pages 2315–2324, 2016.

[27] Xiao Han, Nizar Kheir, and Davide Balzarotti. Phisheye: Live monitoring of sandboxed phishing kits. In *Proc. ACM CCS*, pages 1402–1413, 2016.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[29] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M Voelker, and David Wagner. Detecting and characterizing lateral phishing at scale. In *28th USENIX Security Symposium*, pages 1273–1290, 2019.

[30] Grant Ho, Aashish Sharma, Mobin Javed, Vern Paxson, and David Wagner. Detecting credential spearphishing in enterprise settings. In *26th USENIX Security Symposium*, pages 469–485, 2017.

[31] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean-Michel Picod, and Elie Bursztein. Cloak of visibility: Detecting when machines browse a different web. In *Proc. IEEE S&P*, 2016.

[32] Noman Islam, Zeeshan Islam, and Nazia Noor. A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*, 2017.

[33] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

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

[35] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6(2):8, 2019.

[36] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[37] Hung Le, Quang Pham, Doyen Sahoo, and Steven CH Hoi. URLNet: Learning a URL representation with deep learning for malicious URL detection. *arXiv preprint arXiv:1802.03162*, 2018.

[38] Jehyun Lee, Farren Tang, Pingxiao Ye, Fahim Abbasi, Phil Hay, and Dinil Mon Divakaran. D-Fence: A Flexible, Efficient, and Comprehensive Phishing Email Detection System. In *IEEE EuroS&P*, 2021.

[39] Jehyun Lee, Pingxiao Ye, Ruofan Liu, Dinil Mon Divakaran, and Mun Choon Chan. Building robust phishing detection system: an empirical analysis. *NDSS MADWeb*, 2020.

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

[41] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *30th USENIX Security Symposium*, 2021.

[42] Yun Lin, You Sheng Ong, Jun Sun, Gordon Fraser, and Jin Song Dong. Graph-based seed object synthesis for search-based unit testing. In *European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1068–1080, 2021.

[43] Yun Lin, Jun Sun, Gordon Fraser, Ziheng Xiu, Ting Liu, and Jin Song Dong. Recovering fitness gradients for interprocedural boolean flags in search-based testing. In *International Symposium on Software Testing and Analysis*, pages 440–451, 2020.

[44] Yun Lin, Jun Sun, Yinxing Xue, Yang Liu, and Jinsong Dong. Feedback-based debugging. In *International Conference on Software Engineering (ICSE)*, pages 393–403. IEEE, 2017.

[45] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004.

[46] Christian Ludl, Sean McAllister, Engin Kirda, and Christopher Kruegel. On the effectiveness of techniques to detect phishing sites. In *International Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 20–39. Springer, 2007.

[47] Claudio Marforio, Ramya Jayaram Masti, Claudio Soriente, Kari Kostiainen, and Srdjan Capkun. Evaluation of personalized security indicators as an anti-phishing mechanism for smartphone applications. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 540–551.

[48] Claudio Marforio, Ramya Jayaram Masti, Claudio Soriente, Kari Kostiainen, and Srdjan Capkun. Hardened setup of personalized security indicators to counter phishing attacks in mobile banking. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 83–92, 2016.

[49] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-similarity-based phishing detection. In *Proc. SecureComm*, pages 1–6, 2008.

[50] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proc. IEEE CVPR*, pages 2574–2582, 2016.

[51] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. Phishfarm: A scalable framework for measuring the effectiveness of

evasion techniques against browser phishing blacklists. In *Proc. IEEE S&P*, 2019.

[52] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *29th USENIX Security Symposium*, 2020.

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

[54] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *29th USENIX Security Symposium*, 2020.

[55] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proc. ACM IMC*, pages 478–485, 2019.

[56] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[57] Angelo PE Rosiello, Engin Kirda, Fabrizio Ferrandi, et al. A layout-similarity-based approach for detecting phishing pages. In *Proc. SecureComm*, 2007.

[58] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Aster: An attentional scene text recognizer with flexible rectification. *IEEE Trans. on pattern analysis and machine intelligence*, 41(9):2035–2048, 2018.

[59] Gianluca Stringhini and Olivier Thonnard. That ain't you: Blocking spearphishing through behavioral modelling. In *International Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2015.

[60] Ke Tian, Steve TK Jan, Hang Hu, Danfeng Yao, and Gang Wang. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proc. ACM IMC*, 2018.

[61] Gaurav Varshney, Manoj Misra, and Pradeep K Atrey. A survey and classification of web phishing detection schemes. *Security and Communication Networks*, 9(18):6266–6284, 2016.

[62] Rakesh Verma and Keith Dyer. On the character of phishing urls: Accurate and robust statistical learning classifiers. In *Proc. ACM Conf. on Data and Application Security and Privacy*, 2015.

[63] M Vijayalakshmi, S Mercy Shalinie, Ming Hour Yang, et al. Web phishing detection techniques: a survey on the state-of-the-art, taxonomy and future directions. *IET Networks*, 9(5):235–246, 2020.

[64] Ge Wang, He Liu, Sebastian Becerra, Kai Wang, Serge J Belongie, Hovav Shacham, and Stefan Savage. *Verilogo: Proactive phishing detection via logo recognition*. Department of Computer Science and Engineering, University of California, 2011.

[65] Haijun Wang, Yun Lin, Zijiang Yang, Jun Sun, Yang Liu, Jin Song Dong, Qinghua Zheng, and Ting Liu. Explaining regressions via alignment slicing and mending. *IEEE Transactions on Software Engineering*, 2019.

[66] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-CAM: Score-weighted visual explanations for convolutional neural networks. In *Proc. CVPR workshop*, pages 24–25, 2020.

[67] Jing Wang, Weiqing Min, Sujuan Hou, Shengnan Ma, Yuanjie Zheng, Haishuai Wang, and Shuqiang Jiang. Logo-2k+: A large-scale logo dataset for scalable logo classification. In *Proc. of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6194–6201, 2020.

[68] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[69] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. Cantina+ a feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. on Information and System Security (TISSEC)*, 14(2):1–28, 2011.

[70] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proc. IEEE ICCV*, pages 1369–1378, 2017.

[71] Changhoon Yoon, Kwanwoo Kim, Yongdae Kim, Seungwon Shin, and Sooel Son. Doppelgängers on the dark web: A large-scale assessment on phishing hidden web services. In *Proc. WWW*, pages 2225–2235, 2019.

[72] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *Proc. IEEE S&P*, 2021.

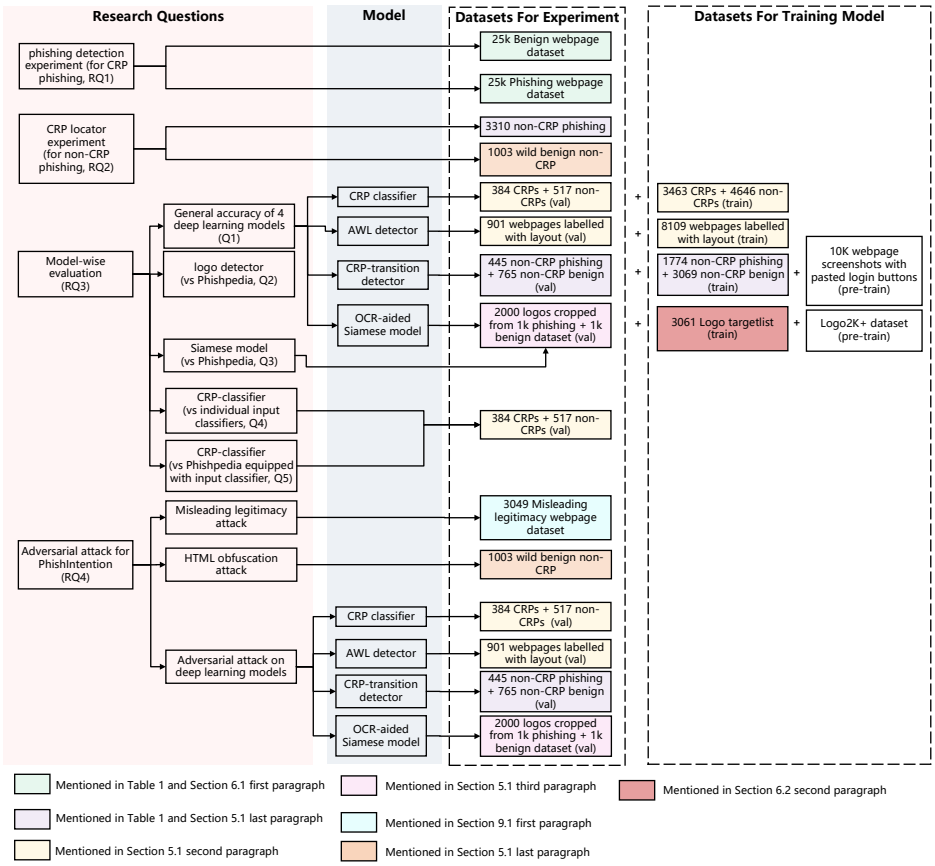


Figure 20: Experiment structure: relations among research questions, models, and datasets

A Appendix

A.1 Hardware Configuration

AWL detection model and CRP-Transition location model are trained with a Faster R-CNN object detector using Detectron2 framework [68]. The OCR-aided Siamese model and CRP classifier are trained using ResNetV2-50 architecture [28, 35]. All deep learning models are trained on an Ubuntu16.04 server with Xeon Silver 4108 (1.8GHz), 128G DDR4 RAM, and NVIDIA Tesla V100 GPU, which takes us 10 hours, 1 hour, 11 hours, and 1 hour to train AWL detector, CRP classifier, CRP-transition locator, OCR-aided Siamese model.

A.2 Dataset Description

As shown in Figure 20, data sources are summarized as below:
Experiment dataset: 25K benign + 25K CRP phishing webpage dataset: See Table 1, from Phishpedia dataset [41].
CRP-Transition locator (hybrid) evaluation set:
(1) 3,310 non-CRP phishing webpage dataset; (2) 1,003 wild benign non-CRP webpage dataset: homepages are collected online from 1K well-known brands.
CRP-Transition locator (Deep Learning part):
(1) 1,210 test set: 445 non-CRP phishing + 765 non-CRP benign, sampled from Phishpedia dataset; (2) 4,843 training set: 1774 non-CRP phishing + 3069 non-CRP benign, sampled from Phishpedia dataset; (3) 10K pre-training set: 10K websites with CRP buttons overlaid
CRP classifier and AWL detector:
(1) 901 test set: 901 webpages labelled with layout and CRP class, sampled from Phishpedia dataset; (2) 8,109 training set: webpages labelled with layout and CRP class, sampled from Phishpedia dataset.
OCR-aided Siamese model:
(1) 2,000 test set: 2,000 logos cropped from 1K benign + 1K phishing; cropped from Phishpedia dataset; (2) 3,061 training set: logo target list is from Phishpedia logo target list of 277 brands; also used as reference logo matching list in real deployment; (3) 167,140 pre-training set: Logo2k+ dataset [67].