Knowledge Expansion and Counterfactual Interaction for Reference-Based Phishing Detection

Ruofan Liu[1,2], Yun Lin[1]*, Yifan Zhang[2], Penn Han Lee[2], Jin Song Dong[2]
Shanghai Jiao Tong University[1] *National University of Singapore*[2]
liu.ruofan16@u.nus.edu, lin_yun@sjtu.edu.cn, zyifan828@gmail.com, leepennhan98@gmail.com, dcsdjs@nus.edu.sg

Abstract

Phishing attacks have been increasingly prevalent in recent years, significantly eroding societal trust. As a state-of-the-art defense solution, reference-based phishing detection excels in terms of accuracy, timeliness, and explainability. A reference-based solution detects phishing webpages by analyzing their domain-brand consistencies, utilizing a predefined reference list of domains and brand representations such as logos and screenshots. However, the predefined references have limitations in differentiating between legitimate webpages and those of unknown brands. This issue is particularly pronounced when new and emerging brands become targets of attacks.

In this work, we propose DynaPhish as a remedy for reference-based phishing detection, going beyond the predefined reference list. DynaPhish assumes a runtime deployment scenario and (1) actively expands a dynamic reference list, and (2) supports the detection of *brandless* webpages with convincing counterfactual explanations. For the former, we propose a legitimacy-validation technique for the genuineness of the added references. For the latter, we propose a counterfactual interaction technique to verify the webpage's legitimacy even without brand information. To evaluate DynaPhish, we constructed the largest *dynamic* phishing dataset consisting of 6344 interactable phishing webpages, to the best of our knowledge. Our experimental results demonstrate that DynaPhish significantly improves the recall of the state-of-the-art approach by 28% while maintaining a negligible cost in precision. Our controlled wild study on the emerging webpages further shows that DynaPhish significantly (1) improves the state-of-the-art by finding on average 9 times more real-world phishing webpages and (2) discovers many unconventional brands as the phishing targets. Our code is available at https://github.com/code-philia/Dynaphish.

*Corresponding author

1 Introduction

Phishing attacks have increasingly emerged in recent years [26, 47–49, 59]. As one of the most prevalent cyber-crimes, phishing attacks are designed to steal users' credentials, resulting in significant financial losses [7, 54], undermining societal reputation [52], and laying a foundation for launching subsequent attacks [36, 56].

Researchers have proposed multiple techniques to detect phishing, including blacklist-based solutions [10, 23, 24] and feature-engineering based solutions [33, 40, 44, 60, 61]. However, in recent years, reference-based solutions [25, 27, 29, 32, 42, 43] have gained popularity due to their superior performance in terms of accuracy, timeliness, and explainability [25, 42, 43].

Equipped with a predefined reference list of domain and brand representations (e.g., logos and screenshots), a reference-based phishing detector [25,27,29,32,42,43] detects phishing webpages based on their domain-brand consistencies. Given a webpage, if it can be matched with a reference brand representation (e.g., PayPal logo) but its domain (e.g., p**o**ypal.com) cannot align with the target domain (e.g., paypal.com), an alarm is reported with the brand-domain inconsistency as the explanation. Compared to blacklist-based and feature-engineering based phishing detectors, state-of-the-art reference-based detectors are reported to detect many more emerging phishing webpages in practice [42, 43].

Although reference-based phishing detectors are effective, they have one critical limitation: they cannot differentiate legitimate webpages from webpages with unknown brands. As a result, both types of webpages are often classified as non-phishing or benign, in an attempt to minimize false positives. Unfortunately, this conservative approach can be vulnerable to phishing attacks that target emerging or regional brands, as attackers frequently launch new phishing campaigns [28].

According to our one-month empirical study [5], phishing campaigns exhibit high levels of dynamism. In a span of just 15 days, the completeness of a predefined reference list can become obsolete by 31.3%. Furthermore, as illustrated in Figure

1, a subset of phishing webpages (approximately 1.5%) either lack brand information or are completely brandless. These factors contribute to a notable increase in false negatives over time, ultimately reducing the effectiveness of reference-based phishing detectors.

In this work, we propose the DynaPhish framework to enhance any reference-based phishing detectors by addressing the aforementioned challenges. Assume that a reference-based phishing detector is deployed to examine a stream of webpages, DynaPhish is designed to (1) actively expand the reference list and (2) detect *brandless* phishing webpages with convincing explanation.

- **Reference Expansion:** We design a reference-validation technique for the crawled legitimate/phishing webpages, extracting the relevant domain and brand representation (e.g., logo) into a verifiable reference in the list. Specifically, we design a *popularity-driven* validation technique to conservatively estimate the legitimacy of a webpage domain with its popularity on the Internet. Our rationale lies in the fact that phishing webpages have no incentive to be published for lasting public societal popularity. The verified pair of domain and brand will be included as a new reference in the list.

- **Behavioral Invariant:** We design a behavior-validation technique as a partial remedy for brandless phishing webpages. We report observed behavioral *invariants* regarding the verifiability of collected credentials on the webpages and the existence of evasive redirection. Based on such invariants, we interact with brandless credential-taking webpages by feeding fake credentials, and record their responses to infer the level of phishing suspicion.

To evaluate how DynaPhish can improve the reference-based phishing detectors, we construct DynaPD (Dynamic Phishing Dataset), the largest live and interactable phishing kits dataset to the best of our knowledge. Existing phishing datasets maintain [42] *static* phishing webpage URLs, screenshots, and HTML code. In contrast, DynaPD works as a clean container hosting 6344 runnable and interactable phishing webpages. Our extensive experiments show that (1) DynaPhish can boost the recall of phishing detection by 28% with a negligible cost in precision and (2) with the increase of the reference list (from 274 to 4177), the state-of-the-art phishing detectors incur little runtime overhead (about 0.5 second) to make the prediction. Moreover, our one-month wild study on the Internet shows that (1) DynaPhish expands reference list by 20 times, (2) DynaPhish allows the state-of-the-art phishing detectors to additionally find 9 times more real-world phishing webpages.

In summary, our contributions lie in the following:

- We challenge the technical assumption of reference-based phishing detectors and demonstrate that relying solely on a predefined reference list may not be adequate to cover the constantly evolving phishing campaigns.

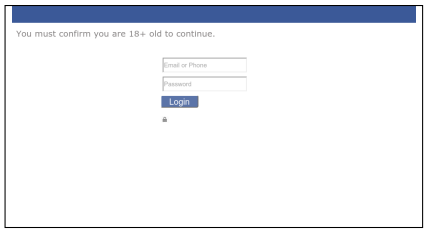


Figure 1: A brandless phishing webpage (URL: https://fb0ik.mydrinkf.xyz/)

- We propose DynaPhish, a systematic remedy (with reference expansion and behavioral invariant) for any reference-based phishing detectors, fixing their inherent limitations on deployment.

- We have constructed and released DynaPD, which stands as the largest dynamic phishing dataset to date. It comprises 6344 and live phishing kits, offering a comprehensive and replicable environment for studying phishing behaviors. This dataset serves as a valuable resource for the development of novel phishing detection solutions and facilitates further empirical studies within the research community.

- Our extensive experiments in both close-world and open-world environments show that DynaPhish is effective and practical, significantly improving the recall of the state-of-the-art phishing detectors with minimal impact on precision and efficiency.

2 Threat Model

We denote a reference-based phishing detector as F, accompanied with a list of references $R = \{ref | ref = (d, r)\}$ where each reference ref is a pair of legitimate domain d (e.g., paypal.com) and its brand representation r (e.g., screenshot and logo). Note that, a brand can have multiple domains and representations. The detector F reports a phishing webpage w along with its targeted brand b if b is kept in the reference list. Otherwise, F reports w as benign.

We assume that the attacker is well aware of the mechanism of F, but the attacker cannot modify F. Additionally, the attacker has full control over the phishing site, enabling them to manipulate the target and webpage design to compromise F in the following manners.

T1: Phishing with Novel Brand. While F effectively provides information on both the phishing target and the result of suspicious webpages, attackers can exploit phishing webpage generators, such as [20], to repeatedly test brands for vulnerabilities. By constructing phishing webpages for new brands absent from the reference list, attackers can elude detection by

Table 1: The functional facilities of reference-based phishing detectors. Different reference-based detectors implement the abstract functions in different ways.

Function	Symbolic Manifestation	Description
$domain(.)$	$w \rightarrow d$	Fetch the domain of a webpage w.
$rep(.)$	$w \rightarrow r$	Extract webpage w's representation. The representation can be screenshot [25, 32] or logo [27, 42, 43]
$match(.,.)$	$(r, r_{ref}) \rightarrow sim$	Compare the similarity between the target representation r and the referenced representation r_{ref}.

F and increase the likelihood of false negatives. This poses a significant threat to the reputation of emerging and regional businesses, particularly in domains like blockchain [3], autonomous driving [1], and cybersecurity [2].

T2: Phishing with Implicit Brand. F detects brand-domain inconsistencies by analyzing the branding intention of webpages. However, attackers can create webpages with implicit brand intention, where the logo is absent (see Figure 1). This deceptive tactic can fool F and lead to the attacker's webpage being misreported as benign.

T3: Phishing with Obfuscated/Adversarial Content. F may utilize HTML heuristics to extract logos and screenshot styles. An attacker can obfuscate the HTML code to make those heuristics ineffective. Furthermore, F may use deep learning models to predict logos and screenshots. An attacker can construct adversarial logos and screenshot images to influence the detection effectiveness of F.

3 Approach

3.1 Overview

We assume that DynaPhish is deployed with reference-based phishing detectors to examine a stream of web pages. We expect that DynaPhish can enhance the reference by adapting to regional and temporal interests. For example, an organization in South Asia can have its customized references (e.g., local popular banks) compared to those in North America. The reference list can also include emerging companies in a timely manner. Neither of these can be easily considered with a predefined reference list.

Figure 2 shows an overview of how DynaPhish works, consisting of two components: (1) a reference-based phishing detector such as VisualPhishNet [25], Phishpedia [42] and PhishIntention [43] and (2) the DynaPhish framework facilitating its effectiveness.

Reference-based Phishing Detection Given a webpage w and a reference list $R = \{ref | ref = (d, r)\}$ where d represents referenced domain and r represents the corresponding brand representation such as logo, we define a reference-based phishing detector F as a function $F : (w, R) \rightarrow b$ ($b \in R \cup \{null\}$) where b is the matched brand reference in R for the webpage w. To clarify, if the matched brand reference is denoted as $b = (d, r)$, we use $b.d$ to represent the domain of the brand and $b.r$ to represent the brand representation. If there is no match between the webpage and any reference in R, we set b to $null$.

F consists of the functional facilities as listed in Table 1, including domain extraction (i.e. $domain(.)$), representation extraction (i.e., $rep(.)$) and representation matching (i.e., $match(.,.)$). Generally, the webpage w is reported as phishing if (1) $b \neq null$ (i.e., the brand representation of w can be matched with that of a referenced brand) and (2) $domain(w) \neq b.d$ (i.e. the domain of w is not consistent with that of the referenced brand). Such domain-brand inconsistency can help us detect and explain the potential phishing attacks.

However, the state-of-the-art [25, 42, 43] does not distinguish the following scenarios, for conservatively controlling the false positive rate:

- w is a **benign webpage**, i.e., $b \neq null$ and $domain(w) = b.d$;
- w targets for an **unknown brand**, i.e., $b = null$. Specifically, given the extracted brand representation (such as logo), there is no reference can be matched in R.

DynaPhish Framework DynaPhish enhances the detection of the state-of-the-art when an unknown webpage (i.e., $b = null$) is detected, consisting of a Brand Knowledge Expansion module and a Webpage Interaction module.

Given a new webpage w, the Brand Knowledge Expansion module utilizes the $domain(w)$ and $rep(w)$ to mitigate the limitation of the predefined R. We design validation techniques (see Section 3.2) to validate the legitimacy of $domain(w)$ via estimating its popularity on the Internet. Then, a new reference $ref = (domain(w), rep(w))$ will be created. In addition, for some non-indexed webpages on the Internet, we use the brand representation $rep(w)$ to further validate its domain legitimacy. If such a domain-representation pair can be extracted and validated, DynaPhish includes it as a reference to enhance R.

If DynaPhish is unable to extract the domain-representation pair, we consider the webpage w to be brandless. In this case, we use the Webpage Interaction module to evaluate its suspiciousness by utilizing our designed behavioral invariants (see Section 3.3.1).

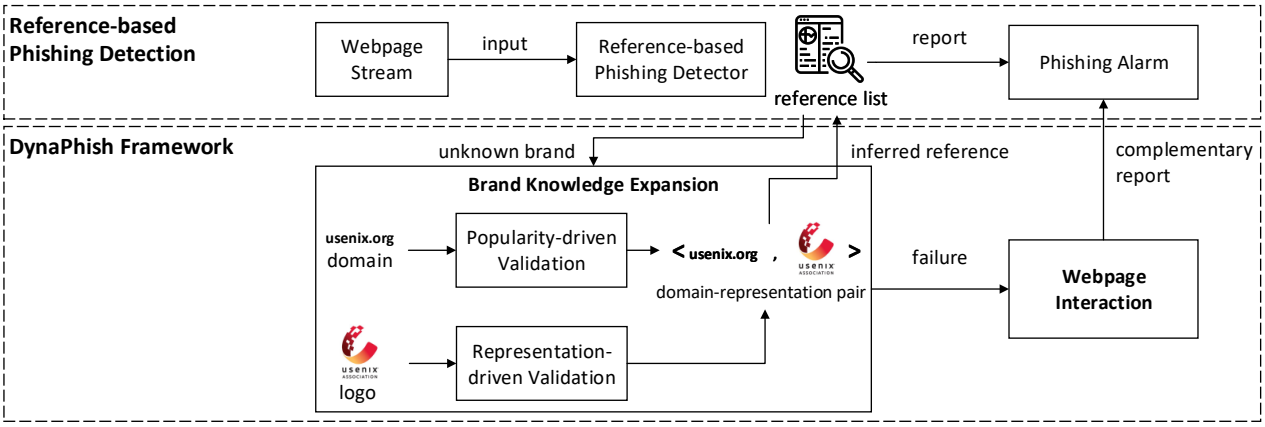


Figure 2: Overview of DynaPhish Framework. The Brand Knowledge Expansion module returns an inferred domain-brand pair. The Webpage Interaction module complements the detection by observing the suspicious behaviors of a webpage.

3.2 Brand Knowledge Expansion

In a reference-based detector, a webpage w is parsed into a domain-representation pair $p = (domain(w), rep(w)) = (d, r)$, where r is unknown with respect to the reference list R. To ensure the trustworthiness of the domain d, we have designed DynaPhish to validate it. To this end, we validate the legitimacy of the domain-representation pair using two hypotheses in sequence:

- **H1**: The webpage w is a *popular* benign webpage.
- **H2**: The webpage w is a phishing webpage, targeting for a *popular* benign webpage.

If we can validate H1, then the domain of w (along with its logo on the homepage) is deemed trustworthy. If we can validate H2, we can identify the real and trustworthy domain that w is attempting to fake. We can then construct a new domain-representation pair in R using this information.

3.2.1 Popularity as Benignity

Phishing attackers have no incentive to publicly expose their links for long-term recognition, as they typically share them privately via email [49]. To evade detection by blacklist-based engines like Google Safe Browsing, these links may persist for a few days [48]. Hence, we argue that the condition "webpage w is popular" is sufficient but unnecessary to classify it as benign (i.e., non-phishing).

Hence, the problem of validating benignity can be reduced to that of validating popularity, which is an easier problem to solve. We define the webpage w as popular if the following criteria are met:

- **C1:** w is indexed by the state-of-the-art search engine (e.g., Google),

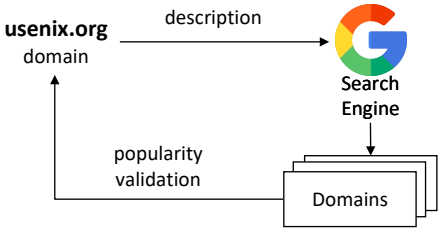


Figure 3: Popularity Validation Overview

- **C2:** w ranks high in the recommendation when its description is fed to search engine, and
- **C3:** w has been alive for a long period of time.

The first and second criteria are devised based on the Page Ranking algorithm [50], which suggests that w could potentially be the most linked webpage under certain topics. The last criterion is formulated based on empirical observations that popular webpages tend to have a longer lifespan. Our empirical study on 30K Alexa webpages [21] indicates that their average lifespan is 10.1 years. Our statistical analysis of 10K recorded phishing webpages reveals that none of them can be classified as popular according to the definition above. For more details, please refer to the DynaPhish website [5].

This problem reduction results in a sound but incomplete solution for DynaPhish, which is generally acceptable since it enables us to cautiously exclude potentially inconclusive and ambiguous references from the reference list.

3.2.2 Popular Benign Hypothesis (H1)

We validate H1 by examining the popularity of the visited webpage. The workflow is illustrated in Figure 3, which comprises a popularity validation process and a consistency vali-

Algorithm 1: Popularity Validation

```
Input  : URL url, retrieval limit k, life span threshold th_life
Output : domain legitimacy is_legitimate
1  domain, tld = domain(url), top-level-domain(url);
   // regards the HTML tag <title>
2  page_title = parse_title(url);
3  W = search_engine_search(domain tld; page_title; k);
4  for w ∈ W do
5      if w.domain == domain and w.tld == tld and
         is_long_alive(w, th_life) then
6          return true;
7  return false;
```

Figure 4: Representation Validation Overview

dation process, as specified in Algorithm 1. Algorithm 1 takes input as the visiting URL url and a retrieval limit k, validating whether the top-k retrieved webpages can (1) contain the domain of url (the first condition in line 5), and (2) satisfy our life-span criteria (the second condition in line 5). The description of w is parsed by extracting its domain and the webpage title. Moreover, the life-span of w is validated in two ways:

- Search Metadata The search engine can return the metadata of each webpage, including the webpage publication date.
- WHOIS Service: The WHOIS service [19] can also provide the historical data of a domain.

3.2.3 Phishing Hypothesis (H2)

We proceed to validate H2 if Algorithm 1 returns false, i.e., $domain(w)$ is not popular. Our validation rationale is that w is phishing if (1) w is not popular and (2) there exists a popular webpage w' with the same brand intention with w.

Figure 4 shows our design for H2 validation. Given a reported brand representation r, such as a logo, we first derive the representation description and feed it into a search engine. To achieve this, we adopt Google's logo detection service [9] to predict the brand name given a logo. Our investigation shows that the service is well capable of detecting the brand names for both text-based logos and non-text-based logos. Subsequently, Google search engine recommends the relevant webpages given the detected brand name. For each of the high-ranking (i.e., top-k) recommendations w' ($w' \in W \setminus w$), we use the representation-extraction function (i.e., $rep(\cdot)$, see Table 1) provided by the reference-based phishing detector to extract its brand representation r'. We then use the representation-matching function (i.e., $match(\cdot,\cdot)$, see Table 1) to check whether r can match r'. If r matches r', we conclude that w is suspicious of phishing and construct $ref = (domain(w'), r')$ as a new reference. If none of the high-ranking webpages have a brand representation that matches r, we conservatively discard w for reference enhancement.

3.2.4 Adaptation to Logo and Domain Variants

The expansion of the reference can result in false positives when a company has multiple domain variants, and false negatives when a company has multiple logo variants. In the case of domain variants, if a company has two domains d_1 and d_2, which share the same brand representation r, a phishing website with domain d_2 may be inaccurately flagged as phishing if only the reference (d_1, r) is present in the reference list. Similarly, in the case of logo variants, if a brand has two versions of logos r_1 and r_2, a phishing website with logo r_2 may go undetected if only the reference (d, r_1) is included in the reference list.

Solution for Domain Variants. During the knowledge expansion phase, prior to the phishing detector raising an alarm indicating inconsistency between a representation r and its domain d, denoted as $\exists b \in R$ such that $match(b.r, r) > th$ but $d \neq b.d$, we conduct additional checks. Specifically, we verify if w satisfies both popularity validation and representation validation (as described in Section 3.2.2 and Section 3.2.3). If the checks are successful, the reference $(domain(w), r)$ is added to the reference list R.

Solution for Logo Variants. To mitigate this issue, we perform an additional search for logo variants using Google Image Search [11] whenever a new reference (d, r) is added. Using a user-defined threshold k, we query Google Image Search with the brand name and the keyword "logo" to retrieve the first k logo images along with their context links. For each image, we verify if the context link corresponds to the domain d. If there is a match, we include the reference set (d, r_k) in the reference list.

3.2.5 Adversary and Maintenance

Next, we discuss potential adversaries and maintenance issue of the reference.

Adversary 1 (Intentional Obsolescence) We consider intentional obsolescence as an adversary to DynaPhish. In this

scenario, an attacker can publish a benign webpage under a newly registered domain and wait for a search engine to index it (which typically takes several months). During this period, DynaPhish may include this domain as a reference if all three criteria (see Section 3.2.1) are satisfied. If the domain is included in R, i.e., the domain-representation pair $ref = (d; r)$, is included, the attacker can then change the webpage content to be phishing. Note that, in this case, $r_{new} = rep(w_{ph}) \neq r$ where r_{new} is the brand representation in the phishing webpage w_{ph}. Assume that we can have $b = (d; r) \in R$ (e.g., via popularity validation) so that $match(r; r_{new}) > th$ and $d \neq d$, the detector can still raise a phishing alarm. Moreover, d can lose its popularity once it presents its phishing intention with w_{ph}.

Adversary 2 (Webpage Injection) Another adversary involves injecting a phishing webpage w_{ph} into a vulnerable but benign website (examples of this can be found in the "Injected Phishing Example" tab on the DynaPhish website [5]). Assuming that the phishing target of w_{ph} is popular, we can find $b = (d; r) \in R$ (e.g., via representation validation) such that $match(r; r) > th$ but $d \neq d$. As a result, the detector will raise a phishing alarm.

Maintenance We consider both the intentional and the unintentional obsolescence as maintenance problem for the reference list. Intentional obsolescence occurs when attackers deliberately change benign content to phishing content, while unintentional obsolescence results from routine updates to webpage design. We design Algorithm 2 to periodically cleanse the reference list. In Algorithm 2, for each reference ref (as a domain-representation pair), we validate its status (updated or obsolete) by checking (1) its potential adversarial intention (lines 2-8) and its potential obsolete status (lines 10-11). We first apply the reference-based phishing detector F on the homepage under the domain of ref to see whether it can be reported as phishing (lines 2-5). If the alarm is raised (line 6), it indicates that two domains conflict to share the same brand intention. Thus, we further validate the popularity and representation for its benignity (line 7). We remove ref from the reference list if it can no longer be validated (line 8). If there is no conflict, we check whether the reported domain-representation pair from the webpage of ref is still the same as that recorded in the reference list (line 10). If not, we update the reference (line 11).

Algorithm 2: Maintenance

```
Input  : reference list Ref, a reference-based phishing
         detector F
Output : updated reference list Ref
1   for ref ∈ Ref do
2       d; r_old = ref;
        // get updated representation
3       r_new = rep(d);
4       Ref' = Ref \ {ref};
5       b = F((d; r_new); Ref');
        // check adversary (intentional obsolescence)
6       if b ≠ null then
7           if d cannot pass popularity validation OR r_new
              cannot pass representation validation then
8               Ref = Ref';
        // check unintentional obsolescence
9       else
10          if r_new ≠ r_old then
11              Ref = Ref ∪ {(d; r_new)};
12  return Ref;
```

3.3 Webpage Interaction

We proceed to the Webpage Interaction module if we cannot identify any brand information from the webpage since the reference comparison is no longer feasible. In this work, we propose two counterfactual behavioral invariants to raise explainable phishing alarms:

- Credential Verifiability Invariant: A webpage w is suspicious of phishing if it cannot verify a randomly generated verifiable credentials (e.g., username or user id).
- Evasion Intention Invariant: A webpage w is suspicious of phishing if it redirects to a different domain while presenting the similar appearance, after the credential submission.

The behavior of a webpage that satisfies these invariants can serve as an explanation for why a phishing alarm was raised. It is important to note that both invariants allow for sound yet incomplete results. For example, a phishing webpage may covertly obtain credentials in the back-end while displaying error messages related to verification, such as an incorrect password warning. However, this approach carries the risk of arousing suspicion among users.

3.3.1 Technical Design

Algorithm 3 shows our design to evaluate webpage behaviors regarding the invariants, which takes a webpage w and a threshold of the number of trials k, and generates the phishing result (i.e., phishing or benign) and its explanation. Algorithm 3 consists of a set of behavioral primitives such as checking whether the webpage requires user credentials (is_require_credential()) identifying the inputs of the webpage (identify_inputs(.)), etc. We generally construct those behavioral primitives with computer vision techniques to minimize the potential risk of HTML code obfuscation.

Algorithm 3 performs the following steps. First, we detect whether the webpage w is a credential-taking webpage (line 1). If not, we check whether any link in w leads to a

Algorithm 3: Web Interaction

```
Input  : A webpage w, trial threshold k
Output : phishing result y, phishing explanation e
1  if !is_require_credential(w) then
2      w ← transit(w);
3      if !is_require_credential(w) then
4          return {benign, ""};
5  I ← identify_inputs(w);
6  I_v ← recognize_verifiable_input(I);
   // no verifiable input
7  if I_v = ∅ then
8      return {benign, ""};
9  b ← identify_submission_button(w);
10 trial number = 1;
11 while trial number ≤ k do
12     w' ← fill_and_submit_form(w, b, I_v);
       // evasion intention invariant
13     if w.domain ≠ w'.domain and w.logo == w'.logo
         then
14         return {phishing, "detecting evasion from
             original domain"};
       // credential verifiability invariant
15     if information_not_verified(w') then
16         return {phishing, "cannot verify fake user
             credential"};
17     trial number ++;
18 return {benign, ""};
```

credential-taking webpage (line 2). If we cannot find such a link, we conclude that w does not require user credentials and report it as benign (lines 3-4). Next, we identify the set of inputs from w as I (line 5) and recognize the inputs requiring verifiable information as I_v (line 6). Verifiable information includes account information that the system can verify, such as usernames, passwords, and emails (we will define this later). If the webpage w does not require any verifiable information, we consider it benign (line 8).

Otherwise, we identify the submission button on the webpage (line 9) and repeatedly input randomly generated fake account information (line 12) to observe whether the webpage response satisfies the credential verifiability invariant or evasion intention invariant (lines 13-16). We perform this observation for k times to minimize the potential for accidental inconsistent responses from the webpage. If the response satisfies the evasion intention invariant (i.e., the webpage redirects to a different domain with the same brand intention), we report w as phishing with the explanation as "detecting evasion of original domain" (lines 13-14). If the response satisfies the credential verifiability invariant (i.e., the webpage can proceed when provided with fake credentials), we report w as phishing with the explanation as "cannot verify fake user credential" (lines 15-16). Next, we explain how we implement our webpage interaction primitives.

Credential Requiring Detection and Webpage Transition (lines 1-2) In this work, we adopt the approach proposed by Liu et al. [43] in their PhishIntention work. They use a deep learning classifier to determine whether a webpage requires user credentials and an object detector to identify links that could potentially lead to a credential-requiring webpage. Since their approach has shown to be effective, we build our DynaPhish system based on their contribution, instead of proposing a new one.

Input Identification (line 5) and Verifiable Input Recognition (line 6) To identify inputs on webpages, we compile all input-taking HTML tags specified in the HTML markup language [12] as potential inputs. We then empirically define 29 types of account information, including usernames, user IDs, emails, credit card information, bank account details, addresses, phone numbers, and more [31]. For more information, readers can refer to our DynaPhish website [5]. Generally, we consider a webpage suspicious of phishing if it cannot verify a fake credential. Our recognition of verifiable information involves two steps: extraction and matching.

- **Extraction**: To extract verifiable information from a given input i on the webpage w, we use a hybrid approach of HTML heuristics and computer vision techniques. The HTML heuristics is designed for efficiency and involves searching for input tag attributes such as placeholder, value, type, and class, with predefined account types. The computer vision technique is designed for robustness against HTML code obfuscation and uses OCR to detect text in the surrounding areas of the inputs. As a result, input i is associated with a text label (e.g., "username").
- **Matching**: We use a fuzz-matching strategy [55] to match the extracted text (e.g., "user name") with our predefined account information descriptions (e.g., "username"). The fuzz-matching strategy is widely used in search engine solutions to retrieve relevant webpages based on user input. Given a predefined account type acc_i and the extracted text l, we transform them into case-insensitive strings acc_i^0 and l^0) and consider them matched if (l^0) has the minimum Levenshtein distance [13] with acc_i^0 compared to the other account types in the list and (2) their Levenshtein distance is smaller than a predefined threshold thd_i.

Technically, we use the state-of-the-art fake credential generation tool, Faker [6], to fill in the detected inputs on webpages.

Submission Button Detection (line 9) We locate the form submission button by training an object detector. We manually labelled the submission buttons on 1495 webpage screenshots and train an object detector to recognize the most likely candidate buttons. We use Faster R-CNN architecture [53] for

designing the object detector and apply Step-ReLU [42, 43] in the model to defend against the potential adversarial attacks [62].

Evaluating Credential Veri ability (line 15) To conservatively report phishing alarm, we de ne that the webpage cannot verify the credential after we feed it with fake credentials if (1) any of the submitted veri able credentials are successfully sent in the network requests and (2) the webpage proceeds to a new page without prompting for the same veri able credentials and (3) no alarm is raised for the use of fake credentials.

4 Experiment Overview

4.1 Research Questions

Our evaluation consist of close-world experiments (RQ1-RQ4) and an open-world experiment (RQ5) to answer the following research questions:

- RQ1 (Overall Effectiveness):What is the overall effectiveness and ef ciency of DynaPhish, comparing to the state-of-the-art approaches?
- RQ2 (Ablation Study): What is the component-wise contribution (i.e., Brand Knowledge Expansion and Webpage Interaction module) on our phishing kit dataset?
- RQ3 (Knowledge Expansion):Whether the included references are accurate and complete, given a large-scale dataset? Whether DynaPhish can deal with the domain variants and logo variants well?
- RQ4 (Adversarial Attacks): What is the robustness of the DynaPhish against adversarial attacks?
- RQ5 (Effectiveness in the Wild):What is the performance of DynaPhish to enhance state-of-the-art reference-based detectors on phishing webpages in the wild?

4.2 Datasets

We prepare the following datasets in the experiments:
Closed-world Dynamic Dataset.In the dynamic dataset, each phishing/benign website is viewable and interactable to evaluate the dynamic features of DynaPhish, such as the Webpage Interaction module. We run 6344 phishing kits on an isolated virtual machine. Section 4.4 further explains how we construct the phishing kits. Furthermore, we collected a comparable number (6309) of top Alexa webpages. We used this dataset to evaluate the overall effectiveness (RQ1), the ablation study (RQ2), and the adversarial attacks (RQ4).
Closed-world Static Dataset.In the static dataset, we have the webpage screenshots and URLs from the benchmark published by [42], consisting of 29,496 phishing and 30,649 benign URLs along with static screenshots. Note that, many of the phishing URLs in this dataset are dead, we track their historical appearance by their screenshots. The larger-scale dataset allows us to evaluate the features of DynaPhish such as the Knowledge Expansion module (RQ3).
Wild-study Dataset. We collect fresh websites from CertStream [4], which provides us with websites that have newly issued or updated TLS certi cates. We crawl 3,000 random fresh websites from CertStream every day. Our wild-study was conducted over 33 days, resulting in a dataset of 99,000 webpages. We used this dataset to evaluate RQ3 (i.e., the performance of DynaPhish on domain and logo variants) and RQ5.

4.3 Performance Metrics

Phishing Detection Effectiveness.The effectiveness of phishing detection is evaluated in RQ1 and RQ2 using precision and recall metrics. Given n ground-truth phishing webpages, a phishing detector reports m webpages as phishing, out of which k are true phishing webpages. We calculate recall as $\frac{k}{n}$ and precision as $\frac{k}{m}$.
Runtime Ef ciency. Ef ciency is evaluated using the rolling median runtime. The median runtime is calculated over every 50 folders processed, generating a smooth plot of runtime over the number of folders scanned.
Knowledge Expansion Effectiveness.We measure the precision and recall for the effectiveness of Knowledge Expansion. Given n ground-truth references to be expanded, if we include m references in the reference list, out of which k references are accurate, we measure recall as $\frac{k}{n}$ and precision as $\frac{k}{m}$.
Logo/Domain Variant Detection Effectiveness.We measure the precision and recall to report an individual logo or domain has variants under its brand. Given n logos/domains has its variants, if we report m logos/domains has variants, out of which k reports are correct, we have the recall $\frac{k}{n}$ and the precision as $\frac{k}{m}$. Note that, we focus on the precision over the recall in the experiment, considering the consequence of false positive is more severe.
Adversarial Robustness.We measure adversarial robustness by evaluating the performance drop of a functionality under adversarial attacks. For deep learning tasks, such as submission button detection on webpage screenshots, we measure performance drop using the gradient-based adversarial attack method and Mean Average Precision (MAP), a widely adopted metric in the object detection community [14]. For the Webpage Interaction module, which detects veri able inputs via HTML analysis, we measure performance drop under HTML obfuscation, using classi cation accuracy as the metric.

Phishing Detection Effectiveness (Wild Study)We measure the recall and precision of phishing detection in the closed-world experiment. Additionally, we compare the number of detected real-world phishing webpages among different reference-based phishing detectors.

4.4 DynaPD Dataset Construction

One challenge in evaluating the interaction nature of DynaPhish is that real-world phishing webpages typically expire within a week [49]. To address this challenge, we constructed the DynaPD dataset, as illustrated in Figure 5. This dataset consists of a container hosting 6344 live and de-weaponized phishing kits.

De nition. A phishing kit comprises all the scripts and resources needed to deploy a phishing website and harvest stolen credentials [28]. In our dataset, the scripts are predominantly written in PHP, HTML, and JavaScript.

Liveness.As shown in Figure 5, we semi-automatically constructed the DynaPD by gathering data from two sources: the Miteru service [15] and a subscribed OpenPhish service [22]. On average, we obtained 30 phishing kits per day. Since the phishing kits are typically written in PHP, we built a virtual machine with an Apache server (XAMPP) to host each kit. Over a 10-month period (from February 2022 to December 2022), we collected a total of 8328 phishing kits, of which 6344 were already interactable or manually repaired to be interactable. These phishing kits cover 567 distinct brands.

De-weaponization.To create a safe phishing dataset for the academic community, we took steps to prevent dataset users' credentials from being sent to phishers. Our analysis showed that 72.11% of phishing kits send credentials via email, 8.66% log the credentials locally, 4.33% use Telegram to send credentials, and 14.91% use both email and Telegram. To de-weaponize each phishing kit, we: (1) replace Telegram-based credential-sending code with email-based credential-sending code and (2) replace the attacker's email with an email address con gurable by the dataset user. We used a taint-based approach to achieve this. Speci cally, we submitted randomly generated information to the phishing webpages, decrypted and monitored the traf c of request content, investigated and modi ed parts of the source code of the phishing kits until no traf c containing such content was observed, and then modi- ed the source code with the user-customized email address. Finally, we provided a set of Selenium-based APIs for the built dataset, including retrieving webpages, capturing screenshots, detecting and lling webpage forms, and more. A trial version of the DynaPD dataset is available at [1][5]

[1]The DynaPD dataset is released based on registration. Thus, we can only provide the trial version regarding anonymous requirement in the conference.

Figure 5: The construction and the design of DynaPD dataset

5 RQ1 (Experimental Effectiveness)

5.1 Setup

In this study, we use the closed-world interactable dataset, as introduced in Section 4.2. We selected Phishpedia [42] and PhishIntention [43] as our baselines, as they have achieved state-of-the-art performance in reporting phishing webpages in both experimental and wild settings. We enhanced these baselines with DynaPhish, resulting in Phishpedia + DynaPhish and PhishIntention + DynaPhish. We used the default reference list published in [43], which consists of 274 references. In our experiments on the DynaPD, 281 out of 567 brands were not included. Moreover, we set the trial threshold $k = 3$ (as shown in Algorithm 3). For more details, please refer to our website [5].

We compare the solutions (i.e., Phishpedia, PhishIntention, Phishpedia + DynaPhish, and PhishIntention + DynaPhish) by their phishing detection performance (as mentioned in Section). To evaluate the potential runtime overhead of DynaPhish, we simulate a practical environment where the number of benign visits is several magnitudes higher than the number of phishing visits. In this experiment, we visit a benign webpage w for $m = d\frac{S}{rank} + 1e$ times, where $rank$ is w's ranking on Alexa. We empirically set S to be 50. Each phishing webpage was visited only once as phishing webpages are typically less visited than benign webpages. Finally, we shuf ed all the visits to the benign and phishing webpages.

5.2 Results

As shown in Table 2, the four solutions were evaluated, and the results indicate that DynaPhish signi cantly improves the recall of PhishIntention and Phishpedia by 28% and 30% respectively, with negligible cost to precision. In general, the brand enhancements made by DynaPhish include many regionally famous brands, such as Australia Post, Intesa Sanpaolo, and Alpha Bank. For more details, please refer to our website [5].

Figure 6 shows the running median of the runtime overhead incurred by DynaPhish on PhishIntention and Phishpedia. Initially, DynaPhish can cause a median runtime overhead of 5.8 seconds per webpage (PhishIntention+DynaPhish) and

Table 2: The performance of the Brand Knowledge Expansion module and the Webpage Interaction module

Solution	Precision	Recall	#Added Ref
PhishIntention	99.85%	40.98%	-
PhishIntention + BKE	99.78%	50.74% (↑ 10%)	3903
PhishIntention + DynaPhish	99.84%	68.63% (↑ 28%)	3903
Phishpedia	99.86%	44.80%	-
Phishpedia + BKE	98.65%	56.26% (↑ 11%)	3903
Phishpedia + DynaPhish	98.97%	74.04% (↑ 30%)	3903
WI	100.00%	43.00%	-

Table 3: Knowledge Expansion Performance

Category	Sub-Category	Recall	Precision
Benign webpages	-	73%	98%
Phishing webpages	Text-based logos (96.4%)	83%	100%
	Non-text-based logos (3.6%)	46%	100%

Figure 6: Runtime Overhead Distribution

5.3 seconds per webpage (Phishpedia+DynaPhish). As the number of references increases, the runtime overhead gradually reduces to 0.5 seconds during webpage processing. Our investigation revealed that the network connection causes the most overhead. Additionally, we observed a signi cant network delay when processing approximately 20% of the webpages in our experimental environment. In consideration of the importance of network stability, we recommend to deploy DynaPhish in two separate working threads. One thread should be dedicated to knowledge construction and expansion, while the other thread should be responsible for phishing detection. This approach will help to prevent any potential blocking of the detection performance. Moving forward, we conduct a thorough investigation into the underlying reasons for the false negatives, readers can refer to our website [5] for details.

6 RQ2: Ablation Study

Table 2 shows the performance of the Brand Knowledge Expansion module and the Webpage Interaction module in enhancing our reference-based phishing detector. In Table 2, we denoted the phishing detector enhanced with only the Brand Knowledge Expansion module as + BKE, and the Webpage Interaction module only as WI. Noted that, the Webpage Interaction module can function independently as a phishing detector. We evaluate the performance of Phishpedia, Phishpedia+BKE, Phishpedia+DynaPhish, PhishIntention, PhishIntention+BKE, PhishIntention+DynaPhish, and WI on DynaPD. Overall, the Brand Knowledge Expansion module signi cantly enhances the recall of PhishIntention and Phishpedia by 10% and 11%, respectively, with a negligible impact on precision. For those phishing webpages where their target brands are not expanded, our investigation shows that the Brand Knowledge Expansion module can be affected by brandless webpages and webpages with obsolete logos. With the support of the Webpage Interaction module, we achieved further performance boost in recall. Note that the Webpage Interaction module can also function as a standalone solution, with a recall of 43.00% and little impact on precision.

7 RQ3-1: Knowledge Expansion

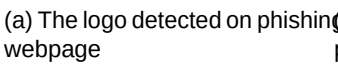
7.1 Setup

To address RQ3, we utilized the benchmark dataset published by [42] to evaluate two key aspects: whether both the brands of benign webpages and the target brands of the phishing webpages can be effectively integrated into the reference list? We evaluate the precision and recall of the expanded brands in benign and phishing webpages individually.

7.2 Results

Table 3 shows the overall performance of our Knowledge Expansion module. Our popularity validation component includes 73% of the Alexa 30K webpages, with a precision of 98%. In contrast, our representation validation component includes 83% of Phish 30K webpages with text-based logos and 46% with non-text-based logos, with a precision of 100%. We will further investigate the false positives and negatives as follows:

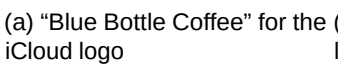
Failure of Popularity Validation on Benign Webpages. Our investigation has revealed two major reasons for failures in expanding brands in benign webpages. Firstly, the benign websites in Alexa 30K can expire. For example, at the time of our experiment, websites such as lkeshuo.com and smart-torrent.org were no longer accessible. Secondly, we observed that some websites take longer to respond to our visit, exceeding our prede ned timeout. Consequently, the Knowledge Expansion module is unable to extract the necessary representations from these webpages. We suggest that practitioners can adjust the timeout to a longer duration if they prioritize completeness of knowledge expansion over timeliness.



(a) The logo detected on phishing webpage

(b) The logo used on benign webpage

Figure 7: Examples of Representations Matching Failures



(a) “Blue Bottle Coffee” for the iCloud logo

(b) “Three” for the Three-UK logo

Figure 8: Performance of Google Logo Detection service: the iCloud and Three-UK logo are mis-reported.

Failure of Representation Validation on Phishing Webpages.As text-based and non-text based logos present different challenges in recognition, we divided the Phish30K dataset into two categories: text-based logos (which represent 96:4% of the dataset) and non-text-based logos (which represent3:6% of the dataset). We observed different reasons for failures in each category. For text-based logos, the Google Logo Detection service allowed us to capture their brand semantics well. False negatives in this category were caused by the logo version in the phishing webpage not matching that in the benign webpage. Nevertheless, the performance of the Google Logo Detection service decreases on non-text logos, leading to more missed brands in Knowledge Expansion. As shown in Figure 8, the Google Logo Detection service may retrieve irrelevant brand names, resulting in missing some brands such as iCloud and Three-UK. Although DynaPhish relies on the functionalities provided by state-of-the-art phishing detectors (e.g. matching two logos) and search engines (e.g. retrieving information from a more extensive information source), its performance is limited by their technical bottlenecks. We will discuss these limitations and their remedies in Section 11.

8 RQ3-2: Logo/Domain Variant Detection

8.1 Setup

We collect ground-truth domain variants from the 99K domains in the wild (as mentioned in Section 4.2). Since it is prohibitively expensive to manually investigate whether every pair of domains is variant, we cluster the domains based on their logo similarity reported by the logo-match implementation in PhishIntention. Then, we manually go through each cluster and decide that there are totally 231 domain variants distributed in 34 brands. We run DynaPhish on those 99K domains to evaluate the precision and the recall of the reported domain variants.

We run our representation validation algorithm against the Fortune 500 [8] company list for the year 2022. In this experiment, we evaluate the precision of the reported logo variants. We also report the average number of logo variants added for each brand.

8.2 Results

DynaPhish achieves the precision of 100% and the recall of 100%, which indicates the effectiveness of our domain-variant search solution (see Section 8.2). In the experiment, the largest group of domain variants lies in the Vivint Smart Home company (https://www.vivint.com/), where every authorized retailer was registered under a different domain but shared the same webpage template.

In addition, our manual investigation of the results of logo variant expansion shows that DynaPhish achieves the precision of100% indicating the effectiveness of the logo-variant search solution (see Section). Examples of logo variants are illustrated in Figure 17.

9 RQ4: Adversarial Attacks

9.1 Setup

In the DynaPhish design, the attack surface mainly lies in (1) the webpage screenshots which can be constructed to fool the deep object detector in DynaPhish to report the submission button (see Section 3.3.1) and (2) the HTML heuristics used by DynaPhish to ll in the fake veri able credentials. Thus, we design two adversaries in this experiment. Note that, we do not evaluate the attack surface of phishing detectors.

First, we used the Dense Adversary Generation (DAG) attack [62] to generate adversarial attacks on the input screenshot. We used mAP (mean Average Precision) [14], a classical performance measurement, to evaluate the object (submission button) detection performance. We manually labeled 1867 Alexa screenshots, sampled from the interactable dataset, as the evaluation dataset. We used 1495 screenshots for training and 372 screenshots for testing.

Second, we generated an adversary for HTML code obfuscation to compromise the HTML heuristics used by DynaPhish for veri able account information (see Section 3.3.1). This adversary consisted of code-to-code obfuscation and code-to-image obfuscation. While we do not guarantee that a real-world attacker would perform similar obfuscation, our obfuscation was designed to ensure that (1) the HTML heuristics in DynaPhish were ineffective under the attack and (2) the webpage still rendered similarly, preserving the webpage semantics. For more details, readers can refer to the DynaPhish website [5] and Appendix (Section A.4).

```
1:
2:<input type="text" ...
3:  id="...", ...,
4:    placeholder="User ID"
5:</input>
```

```
1.<label style="...">User
ID</label>
2:<input type="text" ...
3:  id="...", ...,
4:
5:</input>
```

(a) Previous Version (b) Obfuscated Version

Figure 9: Code-to-code Obfuscation

```
1:
2:<input type="text" ...
3:  id="...", ...,
4:    placeholder="User ID"
5:</input>
```

```
1.<image src="userid.jpg"
...>
2:<input type="text" ...
3:  id="...", ...,
4:
5:</input>
```

(a) Previous Version (b) Obfuscated Version

Figure 10: Code-to-image Obfuscation

9.2 Results

As shown in Table 4, the DAG attack was effective in reducing the mAP measurement by an average of 10%. Moreover, the Step-ReLU defense algorithm worked well to defend the adversarial screenshots (as discussed in Section 3.3.1). Table 5 shows the overall performance of DynaPhish against HTML obfuscation. Overall, the OCR approach complemented the HTML heuristics well enough so that the recall was largely preserved.

10 RQ5: Effectiveness in the Wild

10.1 Setup

We selected Phishpedia [42], PhishIntention [43], Phishpedia+DynaPhish, PhishIntention+DynaPhish, and VirusTotal [18] as the baseline reference-based phishing detectors. To collect emerging webpages, we used CertStream [4], which provided us with webpages that had newly issued or updated TLS certi cates.

We controlled the experiment to crawl 3K randomly fresh websites per day to evaluate the effectiveness of different reference-based phishing detectors. This experiment was conducted for about one month (from 2022-09-02 to 2022-10-04). As a result, 99K fresh webpages were crawled. It is important to note that it is not realistic to achieve the exact recall rate due to the large number of webpages. Therefore, we estimated the precision and recall rates as follows:

We randomly selected 3K webpages and enlisted the help of ve hired interns, each with an average of 1.5 years of cybersecurity experience, to manually label these webpages in an isolated sandbox. If the interns did not agree on a webpage's phishing suspiciousness, we organized a session for all the phishing annotators to reach a consensus. If a consensus could not be reached, we discarded the webpage to be conservative.

Table 4: Testing performance of submission button locator after deep learning adversary

Testing accuracy	mAP IoU 0.5:0.95	mAR IoU 0.5:0.95
Clean	0.746	0.791
After DAG attack	0.630	0.738
After DAG attack with defense	0.751	0.794

Table 5: The robustness against HTML code obfuscation

Dataset	Recall	Precision
Original Accuracy	43.00%	100.00%
Accuracy after Obfuscation	40.40%	100.00%

We take the total number of ground-truth phishing webpages as N, the number of reported and con rmed phishing webpages of an individual solution as N_i, the number of reported phishing webpage of an individual as M_i, the precision is calculated as $\frac{N_i}{M_i}$ and the recall is calculated as $\frac{N_i}{N}$.

10.2 Results

Quantitative Analysis Table 6 shows the overall performance of the ve solutions. In total the Knowledge Expansion of DynaPhish can add over 5K brands into the reference list, starting from the original 274 brands. The augmented reference list generated by DynaPhish boosted its recall by a factor of 10, without compromising its precision. In contrast, Phishpedia enhanced with DynaPhish showed similar boosting performance in recall but had a slight decrease in precision. When compared to PhishIntention, which considers both the brand intention and the credential-taking intention, Phishpedia reports phishing webpages only by the brand intention. In this case, the growth of the reference list can lead to more false positives, which is consistent with the discussion in [43]. As such, we recommend to apply DynaPhish to more advanced reference-based phishing detectors in real-world applications. The other reasons of false positives and negatives are discussed in our website [5]. Next, we report our observed phenomenon different from that in the experimental environment (Section 5).

Observation 1: Unconventional Phishing Targets The huge gap of the recall between F and F + DynaPhish highlights the possibility of phishing campaigns targeting brands that are not traditionally viewed as high-risk targets (such as PayPal or Facebook). Notably, our study reveals a stark contrast in phishing targets when compared to previous literature. As depicted in Figure 11, Cisco emerges as the most frequently attacked brand, with other security and cryptocurrency companies like Sonic Wall [17] and Premint [16] also

Table 6: The performance in the wild study (Precision and Recall are labeled on sampled 3K)

Detector	Precision	Recall	Added References	# Real Phishing
PhishIntention	1.00	0.10	-	127
Phishpedia	1.00	0.05	-	137
PhishIntention + DynaPhish	1.00	0.71	5294	1327
Phishpedia + DynaPhish	0.56	0.79	5294	1366
WI only	1.00	0.17	-	-
VirusTotal	0.01	0.02	-	36

Table 7: The evolving campaigns during the wild study

Period	Top-1 Target	Top-2 Target	Top-3 Target
Day 1 - 5	Microsoft	Facebook	Apple
Day 6 - 10	Cisco	Microsoft	Instagram
Day 11 - 15	Cisco	Microsoft	Sonic Wall
Day 16 - 20	Cisco	Microsoft	Sonic Wall
Day 21 - 33	Cisco	Microsoft	Sonic Wall

Figure 11: The targeted brands of discovered phishing attacks

being targeted. This underscores the potential limitations of using a prede ned reference list, which may restrict the scope of identifying such campaigns.

Observation 2: Dynamics of Phishing Campaigns Table 7 displays the Top-3 phishing targets that were identi ed during different time periods of our study, revealing the dynamic and evolving nature of emerging phishing campaigns. In the rst ve days, Microsoft was the most frequently attacked brand, but was replaced by Cisco from Day 6. Moreover, the second and third most targeted brands changed over time, with Facebook and Apple (in Day 1-5) being replaced by Microsoft and Sonic Wall subsequently from Day 6 to Day 33. It is worth noting that Sonic Wall is a security company, which is not typically viewed as a high-risk target. The use of DynaPhish offers a valuable solution for obtaining a more comprehensive understanding of the dynamics of such campaigns.

Observation 3: The contribution of Web Interaction Module Our evaluation demonstrates that the Web Interaction module is moderately effective in detecting phishing attempts on DynaPD. However, when solely relying on Web Interaction, our recall rate is limited to 0.17 in the wild study. This performance gap may be attributed to the distribution shift between DynaPD and Certstream websites. Generally, the Webpage Interaction module is a conservative phishing detection strategy when the brand intention is not available. It focuses on accuracy instead of completeness, which can further contribute to increasing the evasion costs for phishing attackers.

Overall, DynaPhish can boost the recall of the state-of-the-art phishing detectors in practice. In addition, we empirically observe that the dynamic phishing campaigns are aiming for brands which are not the conventional phishing targets.

11 Limitations and Future Work

Although we have demonstrated that DynaPhish can boost the performance of state-of-the-art reference-based phishing detectors, we acknowledge a few limitations of DynaPhish that highlight areas for the future work.

Performance Limitation and Restriction As demonstrated in our experiments, the boosting performance of DynaPhish is still largely restricted by (1) the information retrieval performance of the search engine and (2) the performance of some basic functionalities, such as representation extraction ($\mathrm{rep}(\cdot)$) and representation matching ($\mathrm{match}(\cdot,\cdot)$). In essence, DynaPhish is a framework that systematically integrates the search engine and the components of a reference-based phishing detector, with their functionalities serving as building blocks. The stronger these blocks, the more effective DynaPhish can be in expanding brand knowledge. However, given the false positives and negatives observed in the experiment, we foresee that the following solutions can be developed:

- A logo-based search engine To the best of our knowledge, the Google service still leads the performance of general information retrieval, especially in terms of text. However, it can still miss expanding some logo information, especially for non-text logos. Therefore, we foresee the development of a logo-based search engine that can overcome these challenges. This can be achieved by using a crawler to search for diverse logos on the internet and constructing a logo-based knowledge graph. With this approach, the logo-retrieval problem can be addressed locally, fundamentally mitigating the information retrieval challenges of the Google service and improving the response time of on-deployment phishing detectors.

- An evolving logo detection/matching model: Although some basic functions, such as representation extraction and representation matching, are initially designed to accomplish reference-based phishing detectors, such as Phishpedia and PhishIntention, they need to be further evolved to support DynaPhish. Specically, the deep learning models used in DynaPhish, such as the object detectors for detecting logos and the Siamese models for comparing logos, require further improvement. Therefore, we foresee the need for an on-deployment model evolution solution that can (1) actively collect more webpage screenshots where the models do not perform well and (2) retrain the models with more informative training samples.
- Limited Behavioural Invariant in Webpage Interaction It is challenging to identify the phishing webpages when their brand intention is not available. In the Webpage Interaction module, we identify two behaviourial invariants: credential verication and evasive redirection. The solution is designed to be conservative, i.e., accurate but not complete, when the brand intention is not available. We foresee that the Webpage Interaction module can play a role to increase the cost of constructing new phishing kits. Moreover, our interaction does not address the advanced cloaking techniques [26, 37, 45]. With client-side cloaking (Figure 12), phishing can bypass many anti-phishing solutions [47, 48].

  In future work, we plan to develop a more robust behavioural-based phishing detection approach by extracting implicit brand intention from webpage layouts. Additionally, we aim to further develop the Webpage Interaction module to address emerging cloaking techniques adopted by phishing attackers.

12 Related Work

Phishing Detection Over the years, phishing detection has undergone several transformations. Initially, it relied on blacklist-based approaches such as Microsoft SmartScreen [24], OpenPhish [22], and Google Safe Browsing [23]. Google Safe Browsing, for instance, accumulates malicious URLs from user reports and crawlers, making it a popular representative of blacklist-based approaches. Alternatively, researchers proposed feature-engineering based approaches [39, 40, 44, 60, 61] that involved extracting features from HTML code, URL, domain, and screenshot, and vectorizing webpages for machine learning models to predict.

However, with the evolution of phishing webpages, both the blacklist and the training phishing dataset have become outdated. To overcome this limitation, reference-based approaches aim to identify suspicious phishing webpages where the brand intention of a webpage does not align with its domain. Medvet et al. [46] and Afroz et al. [27] pioneered this approach by proposing visual webpage representation such as logos to report phishing webpages. With the emergence of deep learning models, Abdelnabi et al. [25], Lin et al. [42], and Liu et al. [43] adopted deep computer vision models to extract brand intention more effectively. However, reference-based detectors also require maintaining a predened set of references, which may become obsolete as phishing campaigns evolve. Therefore, this work serves as a facility to expand and update the reference list, enabling the reference-based approach to remain effective in detecting phishing webpages.

Phishing Kits and Their Analysis The study of phishing kits, which contain the source code for phishing websites, is crucial in understanding how attackers prepare their phishing campaigns. Researchers have discovered that these kits are designed to be evasive in both static and dynamic ways. In terms of static evasion, Cova et al. [30] have found that the critical source code of phishing kits is obfuscated using base64 encoding. As for dynamic evasion, phishing kits often employ cloaking techniques to evade security crawlers. For example, they may generate random URLs [34, 49], detect browser fingerprints [26, 41], detect user interactions [63], and evade two-factor authentication [38, 58]. A systematic and comprehensive cloaking summarization can be referred in Zhang's large-scale study [63]. In addition, empirical studies are also conducted to conrm the evasiveness of phishing websites and investigate their life cycle [28, 35, 37, 49, 51, 57]. To further advance research in this area, we are releasing a new dataset DynaPD of de-weaponized phishing kits for the community. This dataset provides an opportunity to (1) design new phishing detectors based on the interactable phishing websites, (2) gain more insight into the source code of phishing kits, and (3) expand the phishing kit dataset for future research.

13 Conclusion

We present DynaPhish, a solution that enhances reference-based phishing detectors by enabling them to efficiently construct reference lists and detect phishing attempts even on brandless websites through counterfactual interaction. Our extensive experiments demonstrate that DynaPhish significantly reduces false negatives compared to state-of-the-art reference-based phishing detectors.

In the future, we intend to expand our live and interactive phishing dataset, DynaPD, and evaluate the practical performance of DynaPhish by deploying it in an industrial setting.

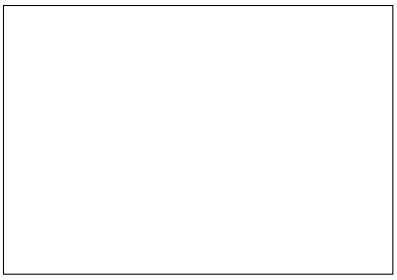


Figure 12: Client-side cloaking in DynaPD dataset

References

[1] 15 autonomous driving companies 2022. https://explodingtopics.com/blog/autonomous-vehicle-startups .

[2] 20 cybersecurity startups 2022. https://explodingtopics.com/blog/cybersecurity-startups .

[3] 50 blockchain companies 2022. https://builtin.com/blockchain/blockchain-companies-roundup .

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

[5] DynaPhish Site. https://sites.google.com/view/dynlaphish-website/ .

[6] Faker. https://faker.readthedocs.io/en/master/ .

[7] Financial losses from phishing, infosec. https://resources.infosecinstitute.com/topic/financial-losses-from-phishing/ .

[8] Fortune 500. https://fortune.com/ranking/fortune500/ .

[9] Google cloud logo detection. https://cloud.google.com/vision/docs/detecting-logos .

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

[11] Google search. https://developers.google.com/custom-search/v1/introduction .

[12] Html: Hypertext markup language. https://developer.mozilla.org/en-US/docs/Web/HTML .

[13] Legenshtein distance. https://en.wikipedia.org/wiki/Levenshtein_distance#cite_note-1 .

[14] map (mean average precision) for object detection.

[15] Miteru. https://github.com/ninoseki/miteru .

[16] Premint. https://www.premint.xyz/ .

[17] Sonic wall. https://www.sonicwall.com/ .

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

[19] Whois. https://en.wikipedia.org/wiki/WHOIS .

[20] zphisher. https://github.com/htr-tech/zphisher .

[21] Alexa. https://www.alexa.com/ , 2020.

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

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

[24] Microsoft SmartScreen. https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-smartscreen , 2021.

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

[26] Bhupendra Acharya and Phani Vadrevu. PhishPrint: Evading phishing detection crawlers by prior profiling. In 30th USENIX Security Symposium (USENIX Security 21), pages 3775–3792, 2021.

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

[28] Hugo Bijmans, Tim Booij, Anneke Schwedersky, Aria Nedgabat, and Rolf van Wegberg. Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection. In 30th USENIX Security Symposium (USENIX Security 21), pages 3757–3774, 2021.

[29] 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.

[30] 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.

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

[32] 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, 3(4):301–311, 2006.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. Catching transparent phish: Analyzing and detecting mitm phishing toolkits. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 36–50, 2021.

[39] 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.

[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] Xu Lin, Panagiotis Ilia, Saumya Solanki, and Jason Polakis. Phish in sheep's clothing: Exploring the authentication pitfalls of browser fingerprinting. In 31st USENIX Security Symposium (USENIX Security 22), pages 1651–1668, 2022.

[42] 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.

[43] Ruofan Liu, Yun Lin, Xianglin Yang, Siang Hwee Ng, Dinil Mon Divakaran, and Jin Song Dong. Inferring phishing intention via webpage appearance and dynamics: A deep vision based approach. In 31st USENIX Security Symposium (USENIX Security 21), 2022.

[44] 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.

[45] Sourena Maroofi, Maciej Korczyński, and Andrzej Duda. Are you human? resilience of phishing detection to evasion techniques based on human verification. In Proceedings of the ACM Internet Measurement Conference, pages 78–86, 2020.

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

[47] 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.

[48] 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.

[49] 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.

[50] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[51] Peng Peng, Chao Xu, Luke Quinn, Hang Hu, Bimal Viswanath, and Gang Wang. What happens after you leak your password: Understanding credential sharing on phishing sites. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pages 181–192, 2019.

[52] James W Ragucci and Stefan A Robila. Societal aspects of phishing. In 2006 IEEE International Symposium on Technology and Society, pages 1–5. IEEE, 2006.

[53] 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.

[54] Ponemon Institute reports. The ponemon 2021 cost of phishing study, 2021.

[55] Peter H Sellers. The theory and computation of evolutionary distances: pattern recognition. *Journal of algorithms*, 1(4):359–373, 1980.

[56] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, et al. Data breaches, phishing, or malware? understanding the risks of stolen credentials. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1421–1434, 2017.

[57] 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.

[58] Enis Ulqinaku, Hala Assal, AbdelRahman Abdou, Sonia Chiasson, and Srdjan Capkun. Is real-time phishing eliminated with {FIDO}? social engineering downgrade attacks against {FIDO} protocols. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3811–3828, 2021.

[59] Cisco Umbrella. Cybersecurity threat trends: phishing, crypto top the list, 2021.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

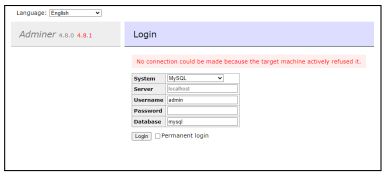


Figure 13: A brandless webpage with fake-information-verification capability

A Appendix

A.1 Runtime Configuration

Choice of hyper-parameters In Table 1, the threshold for representation matching follows the configurations of [42] and [43], which is set to 0.87. For Algorithm 1, we set the retrieval limit k to 10, which is the default number of recommendations per page for Google search. We set the th_{life} to 3 months. The retrieval limit k for Google Image Search (Section 8.2) is also set to 10. Regarding Algorithm 3, the trail threshold k is set to 3 for the experimental dataset, and we reduce it to 1 in the wild study for efficiency considerations. The threshold th_d for Fuzz Matching is set to 1, meaning that we tolerate the OCR to predict only one character incorrectly.

Hardware configuration All experiments are done on an Ubuntu 20.04.3 LTS server with NVIDIA RTX A4000 GPU.

A.2 False Negatives in Experimental Dataset

Credential-Verifiable Brandless Phishing DynaPhish cannot mitigate the false negatives incurred by the brandless phishing webpages with the fake-information-verification capability. Figure 13 shows an example. Its brandless nature makes the knowledge expansion function ineffective. Moreover, the webpage reports an error message such that we cannot decide whether our randomly generated credential is verified or not. As a result, we conservatively report it as non-phishing. This is our limitation by design as we need to conservatively avoid false positives. However, such a brandless phishing webpage makes the users more confused to provide credentials, which makes the phishing attack inefficient.

Obsolete Logo in Phishing We observe that some phishing webpages can use obsolete logos which makes our representation validation (see Section 3.2.3) ineffective. Figure 14 shows an example where the phishing webpage uses an obsolete logo of America First, a loan company in the United States but not that famous globally. Our representation validation technique extracts the description of "America First" from the logo and retrieves the legitimate domain (i.e., https://www.amerfirst.org/) accordingly. However, the logo of the company has been updated (see Figure 15), the mismatch makes us not raise an alarm.