PhishDecloaker: Detecting CAPTCHA-cloaked Phishing Websites via Hybrid Vision-based Interactive Models

Xiwen Teoh[1,2], *Yun Lin*[1,*] *Ruofan Liu*[2], *Zhiyong Huang*[2], *Jin Song Dong*[2]
Shanghai Jiao Tong University[1], *National University of Singapore*[2]
xiwen@nus.edu.sg, lin_yun@sjtu.edu.cn, liu.ruofan16@u.nus.edu, dcshuang@nus.edu.sg, dcsdjs@nus.edu.sg

Abstract

Phishing is a cybersecurity attack based on social engineering that incurs significant financial losses and erodes societal trust. While phishing detection techniques are emerging, attackers continually strive to bypass state-of-the-arts. Recent phishing campaigns have shown that emerging phishing attacks adopt CAPTCHA-based cloaking techniques, marking a new round of cat-and-mouse game. Our study shows that phishing websites, *hardened* by CAPTCHA-cloaking, can compromise all known state-of-the-art industrial and academic detectors with almost *zero* cost.

In this work, we develop PhishDecloaker, an AI-powered solution to *soften* the shield of the CAPTCHA-cloaking used by phishing websites. PhishDecloaker is designed to mimic human behaviors to solve the CAPTCHAs, allowing modern security-crawlers to see the uncloaked phishing content. Technically, PhishDecloaker orchestrates five deep computer vision models to detect the existence of CAPTCHAs, analyze its type, and solve the challenge in an interactive manner. We conduct extensive experiments to evaluate PhishDecloaker in terms of its effectiveness, efficiency, and robustness against potential adversaries. The results show that PhishDecloaker (1) recovers the phishing detection rate of many state-of-the-art phishing detectors from 0% to up to on average 74.25% on diverse CAPTCHA-cloaked phishing websites (2) generalizes to unseen CAPTCHA (with precision of 86% and recall of 69%), and (3) is robust against various adversaries such as FGSM, JSMA, PGD, DeepFool, and DPatch, which allows the existing phishing detectors to achieve new state-of-the-art performance on CAPTCHA-cloaked phishing webpages. Our field study over 30 days shows that PhishDecloaker can help us uniquely discover 7.6% more phishing websites cloaked by CAPTCHAs, raising alarm of the emergence of CAPTCHA-cloaked features in the modern phishing campaigns.

[*]Corresponding author

1 Introduction

Phishing attacks cause enormous financial losses and undermine societal trust. In recent years, the number of phishing attacks has grown by over 150% per year [1]. To mitigate these consequences, researchers have proposed various phishing detection solutions [12,15,34–37,42] to report and explain diverse zero-day phishing websites. While those solutions can be effective against the phishing website, their effectiveness is largely based on the assumption that *a security crawler can access the phishing content of the websites*. Unfortunately, in the new round of phishing campaigns, a growing body of evidence [46,64] has shown that the assumption is less likely to hold true due to the emergence of CAPTCHA-based cloaking techniques.

Cloaking is an evasion technique increasingly adopted by phishing attackers to display different content to security crawlers and human victims [64]. Attackers can deploy either server-side or client-side cloaking for their phishing webpages. Server-side cloaking checks human visits by analyzing HTTP requests from the server end and deny visits from certain IP addresses and User-Agents [32]. On the other hand, client-side cloaking checks human visits by analyzing the runtime browser behavior, including cookies, canvas fingerprints, and WebGL capabilities [13,64]. In recent years, researchers and security engineers have proposed remedies such as simulating a human-mimic HTTP header (to address server-side cloaking) [32] and forcing the execution of JavaScript code used in cloaking (to address client-side cloaking) [64]. However, CAPTCHA-based cloaking, being a novel phishing-cloaking technique, can easily nullify those anti-cloaking efforts.

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) was initially developed as challenge-response authentication to limit the abuse of web crawling. CAPTCHA validates a human visit with the interaction between the client and the server. On the client side, the website prompts a CAPTCHA challenge, such as picture selection and text recognition, to collect the challenge response. On the server side, the challenge response is vali-

Listing 1: Embeded CAPTCHA Code in HTML File.

```html
<html>
...
<!-- CAPTCHA library>
<script src="https://js.hcaptcha.com/1/api.js"
        async defer></script>
...
<!-- embedded CAPTCHA div-tag>
<div id="cloaking">
    <form id="form" method="post">
    <div class="h-captcha" data-sitekey="..."
        data-callback="submitForm" />
    <input type="hidden" value="hcaptcha" name="
        captchaType" />
    </form>
</div>
...
</html>
```

dated against the ground-truth answer. By this means, neither the HTTP-request modification nor the Javascript force execution technique can bypass this validation. CAPTCHA can effectively serve as a cloaking technique in three aspects:

- **False Sense of Legitimacy:** CAPTCHAs are widely used on legitimate websites, allowing a phishing website with a prompted CAPTCHA challenge to often maintain its plausibility without arousing suspicion.

- **Low Deployment Cost:** Any website can conveniently integrate a CAPTCHA service by calling its API (see Listing 1). Therefore, it is not difficult for phishing attackers to automatically generate phishing kits equipped with CAPTCHA cloaking. Additionally, there are many free CAPTCHA services (e.g., reCAPTCHA v2) available, resulting in almost zero cost to *harden* phishing websites with a CAPTCHA.

- **Hard to Bypass:** Due to CAPTCHA's client-server architecture, it is non-trivial for modern security crawlers to automatically bypass it.

Recent studies have shown that phishing attackers are adopting CAPTCHA as a novel cloaking technique [14, 37, 41, 46, 59, 64]. The number of CAPTCHA-cloaked phishing websites has increased almost tenfold from 55,447 on January 2023 to 524,344 on June 2023. [10]. Furthermore, our empirical study (see Section 2) shows that *none* of the publicly available industrial phishing detection engines or academic state-of-the-art models can detect a CAPTCHA-cloaked phishing website.

In this work, we propose PhishDecloaker as the first step to address the CAPTCHA-based cloaking problem. PhishDecloaker is designed to simulate human behaviors in order to solve the CAPTCHA in an interactive manner. Technically, PhishDecloaker employs five types of deep computer vision models across three stages: detection, recognition, and solving. Specifically, PhishDecloaker begins by detecting the presence of a CAPTCHA, formulating it as an object detection problem [35,36,59] on a webpage screenshot. Then, PhishDecloaker recognizes the type of CAPTCHA (by treating it as a metric learning problem) so that it can schedule a follow-up solving plan. This three-stage design allows us to flexibly extend PhishDecloaker to solve new types of CAPTCHA and maintain its performance even on out-of-distribution CAPTCHAs. Our implementation of PhishDecloaker supports reCAPTCHA v2, hCaptcha, slider CAPTCHA, and rotation CAPTCHA, covering 98.9% of the CAPTCHA market share [2].

We conduct extensive experiments to evaluate PhishDecloaker regarding its effectiveness, efficiency, and robustness against potential adversaries. The results show that PhishDecloaker (1) recovers the phishing detection rate of many state-of-the-art phishing detectors from 0% to an average of 74.25% on diverse CAPTCHA-cloaked phishing websites and (2) generalizes to unseen CAPTCHA (with average precision and recall of 86% and 69%), and (3) is robust against adversarial attacks such as FGSM, JSMA, PGD, DeepFool, and DPatch. Furthermore, our field study over 30 days on the emerging real-world websites with different decloaking techniques show that PhishDecloaker allows us to detect 7.6% more phishing websites cloaked by CAPTCHAs (66 out of 869 phishing websites).

In summary, this work makes the following contributions:

- We develop PhishDecloaker, a hybrid deep-vision system to detect, recognize, and solve diverse CAPTCHAs. This system supports mainstream CAPTCHAs and is designed to be extensible for new types of CAPTCHAs. To the best of our knowledge, our work is the first to address CAPTCHA-cloaking for phishing detection.

- We deliver Cloaken, a CAPTCHA-based hardening framework, which allows us to automatically cloak a phishing kit with CAPTCHAs and deploy it through randomly generated URLs. We implement this cloaking technique on top of the DynaPD dataset[1] [37].

- We deliver PhishDecloaker as a tool integrated with existing SOTA phishing detectors Phishpedia [35] and PhishIntention [36], enhancing their capabilities to detect zero-day phishing websites.

- We conduct extensive experiments to evaluate PhishDecloaker. Our results show that PhishDecloaker effectively decloaks phishing websites found in the wild. Furthermore, PhishDecloaker is robust against out-of-distribution CAPTCHAs and adversarial attacks.

Given the space limit, more details of PhishDecloaker are available at [9].

[1]The DynaPD dataset provides more than 6,000 phishing kits for security researchers to interact with.

2 An Empirical Study of Anti-Phishing Entities against CAPTCHA-Cloaking

In this section, we conduct an empirical study to answer the question: *what is the performance of the state-of-the-art anti-phishing solutions in detecting CAPTCHA-cloaked phishing?*.

To answer the question, we design a phishing hardening framework, Cloaken, to automatically cloak phishing kits with CAPTCHAs. Technically, Cloaken functions as a reverse proxy that blocks visitors with a CAPTCHA page from a selection of templates (e.g., hCaptcha, reCAPTCHA v2). Once the visitor solves the CAPTCHA, Cloaken verifies the submitted challenge and reveals the phishing content.

Phishing Detection Service. We select 2 popular URL blacklists: Google Safe Browsing (GSB) and Microsoft Defender SmartScreen for evaluation. GSB is used by Chrome, Firefox, and Safari web browsers, accounting for 81.36% of desktop and 90.17% mobile users worldwide. On the other hand, SmartScreen protects Edge and accounts for 12.75% of desktop users. We also include VirusTotal (VT) [52], the world's largest threat corpus with 92 integrated phishing detectors.

URL Configuration. We prepare 5 cloaking types of phishing websites (t_p): no cloaking (baseline) and 4 *CAPTCHA-cloaked* types (i.e., reCAPTCHA v2, hCAPTCHA, GeeTest Slide, and Rotation). They are choosen based on their market share [54]. We call a pair of detection service and cloaking type, (d, t_p), as a configuration. For each cloaking type t_p, we generate k random unique URLs and submit it to d for analysis. In this study, we let k be 100. Therefore, there are 500 URLs submitted to 3 different detection service.

URL Submission. To report each of our phishing URLs, we submit them to GSB via its online submission portal and to VirusTotal via its API. Since SmartScreen receives data from other anti-phishing entities, including Microsoft's own internal cybersecurity ecosystem, we mass-submit emails with the phishing URLs to an Outlook account with Microsoft Defender Safe Link activated.

URL Monitoring. We monitor every day if any of the URLs corresponding to a configuration (e.g., d-t_p pair) are reported as phishing. If it is, it means that a service d can penetrate a cloaking type t_p. We monitor GSB through its Lookup API and VT by requesting a new URL scan and reviewing the resulting analysis report. We monitor SmartScreen by loading the URLs onto a non-headless Edge browser using Playwright and check if SmartScreen's block page shows up.

Results. Table 1 shows our results. All baseline sites are blacklisted within 24 hours, whereas all CAPTCHA-cloaked sites remain undetected for 7 days and counting, which is more than sufficient for a successful phishing campaign [47, 49][2]. Overall, none of the selected phishing detectors can reveal CAPTCHA-cloaked phishing, further motivating us to design PhishDecloaker. We will discuss the ethical concerns on this experiment in in Section 6.

[2] We reported our findings to VT, GSB, and SmartScreen. GSB (Google) and SmartScreen (Microsoft) have acknowledged our report and are investigating the potential solution.

3 Threat Model

Assume that there is a set of phishing detectors $\mathcal{D} = \{d_1, d_2, ..., d_n\}$, where each detector d_i ($i = 1, 2, ..., n$) needs to access the content of a webpage $w \in \mathcal{W}$ for phishing analysis. Each detector d_i can be modelled as a function $d_i(.) : \mathcal{W} \rightarrow \{0, 1\}$ where $\mathcal{W}$ is the set of webpages. Specifically, a detector $d_i(.)$ maps a webpage $w \in \mathcal{W}$ to a boolean value where $w = 0$ or 1 indicates that w is benign or phishing.

An attacker can equip their phishing website w_p with a CAPTCHA instance $c \in \mathcal{C}$ where $\mathcal{C}$ is the set of CAPTCHA instances under pre-defined CAPTCHA types (e.g., reCAPTCHA, hCaptcha, and GeeTest). Equipped with a human-authentication challenge c, the attacker can render a new webpage $w'_p \leftarrow c \oplus w_p$ so that $\forall d_i \in \mathcal{D}, d_i(w'_p) = 0$, where $\oplus$ is an operation to render the CAPTCHA on top of the webpage w_p. The prepared CAPTCHA sets share the following features:

- **Diverse Types of CAPTCHAs**. The attacker can adopt diverse types of CAPTCHAs, including commercial versions (e.g., reCAPTCHA and hCaptcha) and open-source versions. Additionally, we assume that the attacker can customize their own implementation of well-known CAPTCHA challenges, for example, with a similar appearance to reCAPTCHA and hCaptcha.
- **Code Obfuscation**. The CAPTCHA can have its partial execution on the client side, for example implemented by JavaScript code. We assume that the attacker can adopt code obfuscation techniques [11, 58] to modify the underlying code or structure of CAPTCHAs while preserving the same appearance and functionality inside a browser.
- **Adversarial Images**. An attacker may introduce noise and distortion into CAPTCHA images [30, 50, 56]. This can make it difficult for deep learning models to extract the relevant features and classify the CAPTCHA accurately, as the modified CAPTCHA may no longer conform to the expected patterns or features used for classification.

Given the above threat model, we adopt a vision-based solution to detect and classify CAPTCHAs. In other words, our solution (1) does not rely on frontend code analysis and (2) must be robust against out-of-distribution CAPTCHAs and adversarial attacks such as noise and distortion.

4 Approach

Overview. Figure 1 provides an overview of PhishDecloaker. Given a suspicious webpage w_p with cloaking potential, instead of feeding w_p into a phishing detector, PhishDecloaker tries to remove its "cloak" by looking at and interacting with

Table 1: Empirical results on performance of SOTA industrial phishing detectors in revealing CAPTCHA-cloaked phishing sites.

Category	Phishing Detection Service	URLs Blacklisted / URLs Submitted				
		Baseline	reCAPTCHA v2	hCaptcha	GeeTest Slide	Rotation
API-based	VirusTotal *(incl. 92 phishing detectors)*	100 / 100	0 / 100	0 / 100	0 / 100	0 / 100
Browser-based	Google Safe Browsing	100 / 100	0 / 100	0 / 100	0 / 100	0 / 100
Browser-based	Microsoft Defender SmartScreen	100 / 100	0 / 100	0 / 100	0 / 100	0 / 100

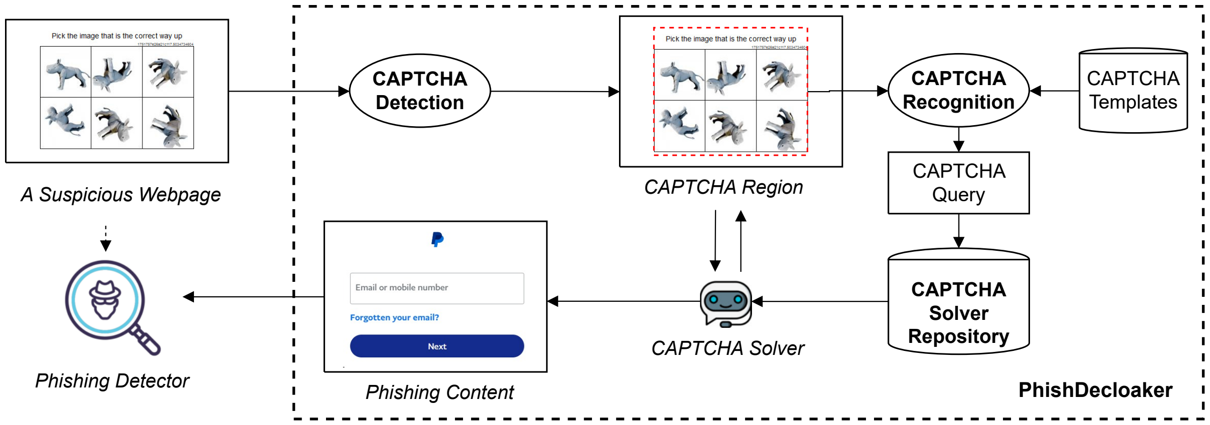


Figure 1: System Design of PhishDecloaker. PhishDecloaker is designed to remove the "cloak" of a CAPTCHA-cloaked phishing webpage, by detecting, recognizing, and solving the CAPTCHA challenges.

w_p. Technically, PhishDecloaker operates on the screenshot of w_p to bypass potential code obfuscation, which involves three steps:

Step 1. CAPTCHA Detection (Section 4.1). We identify the CAPTCHA instance on the webpage by formulating it as an object detection problem in computer vision. We denote the detected CAPTCHA as c.

Step 2. CAPTCHA Recognition (Section 4.2). With our prepared database of CAPTCHA templates, we determine the type of the CAPTCHA c, t_c, by matching the CAPTCHA instance c with its best fit in the template database through a learned OCR-aided Metric Learning network. The solution is designed in a similar way as a face recognition problem in computer vision. By this means, PhishDecloaker provides an extensible CAPTCHA recognition framework to flexibly including new CAPTCHA types.

Step 3. CAPTCHA Solving (Section 4.3). PhishDecloaker is further equipped with an arsenal of CAPTCHA solvers. Given the CAPTCHA type t_c, we formulate it as a query to find the most appropriate CAPTCHA solver. The found solver interacts with the webpage w_p to solve the challenge. This interaction can be repeated multiple times to increase the success rate of anti-phishing crawlers. PhishDecloaker provides such an extensible design to integrate new CAPTCHA solvers in the CAPTCHA solver repository.

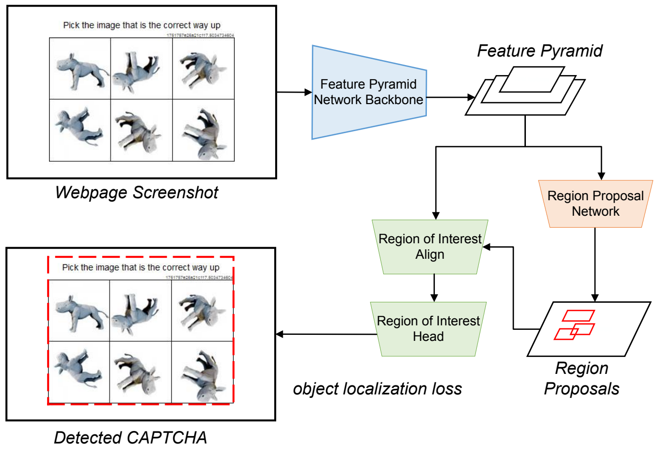


Figure 2: Model architecture of CAPTCHA detection model, consisting of multi-stage processes

4.1 CAPTCHA Detection

Given a webpage screenshot, denoted as $\mathcal{S}$, as input, the CAPTCHA detection model generates object proposals $\mathcal{C}(\mathcal{S}) = \{t | t = \langle x, y, w, h \rangle\}$. As showed in Figure 2, these proposals (in red dashed rectangle) consist of bounding boxes that contains the CAPTCHA region.

Figure 2 illustrates our model design. We employ an Object Localization Network (OLN) [33], which is a two-stage network comprising a Region Proposal Network (RPN) stage and a Region of Interest (RoI) stage. Given a webpage screenshot, the backbone network (Feature Pyramid Network) transforms the webpage into a feature pyramid $\mathcal{F} = \{f | f = k \times k, f \in R^2\}$ where $k = \frac{W}{4}, \frac{W}{8}, \frac{W}{16}....$ Each element in $\mathcal{F}$ is a feature map in the form of a $k \times k$ matrix, and W is the maximum of webpage width and webpage height. Each feature map captures the spatial features of the webpage screenshot in different granularity. Then, the feature pyramid is fed into the Region Proposal Network, generating initial location proposals for the foreground object (i.e., CAPTCHA). These proposals undergo further refinement in the Region of Interest (RoI) network to yield the final bounding boxes.

Different from a conventional object detection model [53] where objects are both detected and classified, we customize our CAPTCHA detector to have only detection functionality. Technically, instead of a training objective covering (1) object location on x, y, w, and h and (2) object classification, we train our CAPTCHA detector with a sole focus on object location. The customization allows our model training process to focus on a single optimization objective. This is useful considering that the CAPTCHA containers can be very diverse, which can include (1) task instructions for solving the CAPTCHA, (2) the challenge body with visual elements, and (3) user interaction buttons for controlling and engaging with the CAPTCHA. They can take various forms and styles, which may encompass distorted text, images, specific object clicks, or even behavioral cues such as slider dragging.

4.2 CAPTCHA Recognition

The CAPTCHA recognition model is designed to map a test CAPTCHA instance c to its best fit in a set of prepared CAPTCHA templates $\mathcal{C}_t$ where each element in $\mathcal{C}_t$ is a representative CAPTCHA instance of a CAPTCHA type. Our CAPTCHA recognition model consists of a feature extractor mapping an object proposal (i.e., a CAPTCHA instance) to a feature vector, i.e. $f_\theta(.) : \mathcal{C} \rightarrow \mathbb{R}^n$. We denote the type function $type : \mathcal{C} \rightarrow \mathcal{T}$ which returns the type of a CAPTCHA instance, where $\mathcal{T}$ is the set of CAPTCHA types in $\mathcal{C}_t$. Then, we can select the best fit c^* of a test CAPTCHA instance c by

$$c^* = \arg\max_{c_t \in \mathcal{C}_t} cos(f_\theta(c), f_\theta(c_t))$$

Given a threshold th, we can decide the type of c by $type(c^*)$ if $cos(f_\theta(c), f_\theta(c^*)) > th$. To learn f_θ, we address the following challenges:

- **Multi-modal representation learning:** A CAPTCHA challenge contains multi-modal information, including the challenge description in both plain text and images.
- **Intra-type diversity:** Challenges within the same CAPTCHA type can differ significantly due to various service vendors or updates to the CAPTCHA pool.

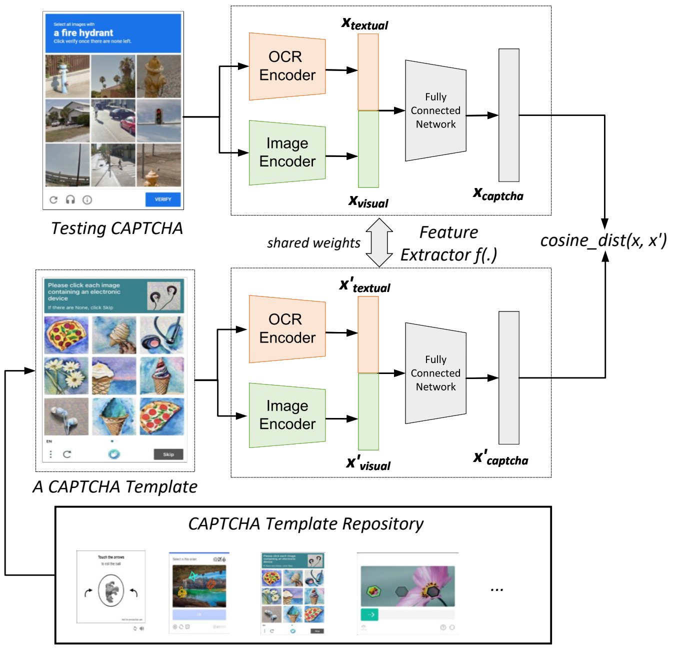
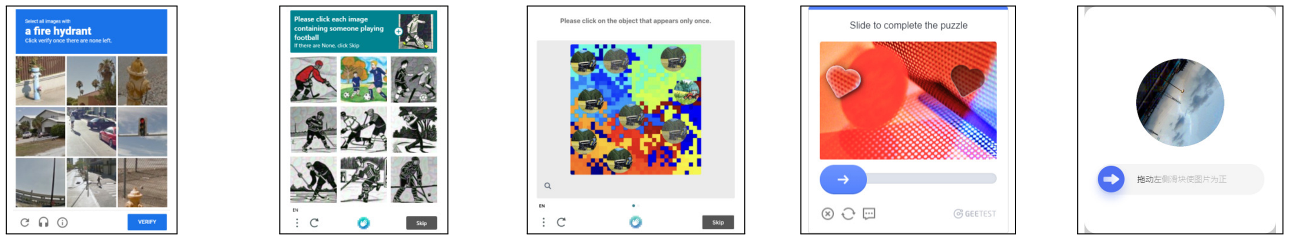


Figure 3: OCR-aided Metric Learning Model. The CAPTCHA recognition system comprises a feature extractor $f(.)$ and a recognition head. The extractor maps a CAPTCHA image to a low-dimensional embedding $\mathbf{x}_{captcha}$. The recognition head matches the testing CAPTCHA with CAPTCHA templates using cosine distance.

- **Inter-type generalization:** As CAPTCHA technology evolves, new types emerge. The model must be easily adaptable to new types of CAPTCHAs.

To incorporate both textual and visual features into the representation, we introduce a dual-branch architecture for our feature extractor $f_\theta(.)$ (See Figure 3). The architecture consists of: (1) a text encoder, pre-trained on an Optical Character Recognition (OCR) task, and (2) an image encoder, pre-trained on an image classification task. Both encoders takes the CAPTCHA image as input, and produces the respective embeddings $\mathbf{x}_{visual}$ and $\mathbf{x}_{textual}$. These two branches capture distinct yet complementary information. The OCR-based encoder focuses on character-indicative features essential for understanding task instructions. In contrast, the image encoder identifies salient visual patterns, capturing the CAPTCHA's layout and design. A fully-connected projection layer is added to fuse the two modalities with additional non-linearity: $\mathbf{x}_{captcha} = \sigma(W^T[\mathbf{x}_{visual} \oplus \mathbf{x}_{textual}] + \mathbf{b})$.

With the feature extractor, we design the recognition head. A straightforward approach involves adding a Softmax activation and using conventional Cross-entropy Loss for model fitting. However, this approach tends to overfit to observed samples, especially when the training set is small, leading to inaccurate predictions for new variants within known categories. Additionally, this approach lacks the flexibility to


a fire hydrant
VERIFY
Please click each image containing someone playing football
Skip
Please click on the object that appears only once.
Skip
Slide to complete the puzzle
GEETEST
拖动左侧滑块使图片为正

(a) reCAPTCHA v2 (b) hCaptcha (Variant 1) (c) hCaptcha (Variant 2) (d) GeeTest Slider (e) Baidu Rotation

Figure 4: Examples of CAPTCHA challenges of different types.

accommodate new CAPTCHA types during runtime, as the number of classes must be predefined before inference.

In this work, we employ a deep metric learning solution to address the challenges outlined earlier. The model aims to learn an embedding space that accurately captures semantic similarities between images. It is trained on pairs of samples, denoted as "positive pair" if they belong to the same class and "negative pairs" if they come from different classes. A loss function ensures that positive pairs are closer in the embedding space than negative pairs. Specifically for CAPTCHAs, our goal is to cluster those of the same type together. During inference, we input the test CAPTCHA along with a set of CAPTCHA *templates* from different classes of CAPTCHAs. We then rank the distances to identify the closest class as the final prediction.

In this manner, we effectively tackle the aforementioned challenges: For the *intra-type diversity* issue, the pairwise training paradigm enables the model to be more sensitive in distinguishing "variations within the class" from "variations relative to other classes". Given an unseen sample from a known class, the model is more inclined to treat it as a variant of a known class rather than a novel class of CAPTCHA. For the *inter-type generalization* issue, accommodating new CAPTCHA types is straightforward: new CAPTCHAs can be easily added to the template database, serving as reference points for future queries.

Technically, during training, we freeze the textual branch and fine-tune all other remaining modules using Sub-center ArcFace loss [21]:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{e^{s\cdot(\cos(\theta_{y_i}+m)-1)}}{e^{s\cdot(\cos(\theta_{y_i}+m)-1)}+\sum_{j=1,j\neq y_i}^{C}e^{s\cdot\cos(\theta_j)}}\right) \tag{1}$$

In Equation 4.2, we learn a set of parameters representing the embedding centers for each class. The embedding feature i and the center for its ground truth class y_i are considered a "positive pair", while i and the center for another class j form a "negative pair". θ_{y_i} is the angle between the positive pair, and θ_j is the angle between the negative pair. This loss function encourages CAPTCHA embeddings to be close to their respective class centers and distant from irrelevant ones. Additionally, to address class imbalance, as some CAPTCHA types are more common than others, we assign class weights to the loss, calculated as $\frac{1}{\log(n_c)}$, where n_c is each class's frequency in the training set.

4.3 CAPTCHA Solvers

The types of challenges presented by CAPTCHAs are diverse, as shown in Figure 4. Each type may require a unique skill set, such as object recognition, visual question answering, pattern matching, or orientation identification. To address the problem, we develop an arsenal of CAPTCHA solvers for each supported CAPTCHA type. For some CAPTCHA types (e.g., reCAPTCHA), we adopt the state-of-the-art solvers; while for other important CAPTCHA types (e.g., rotation) where no solver is available, we develop our own AI-powered solving solutions. We do not claim contribution in solving a particular CAPTCHA type as our system is extensible to new CAPTCHA solvers. We support four types of CAPTCHA solvers as follows.

reCAPTCHA v2 Solver. Google reCAPTCHA v2 is the most prevalent type of CAPTCHA among the Top 1 Million Sites [2]. We solve the reCAPTCHA v2 challenges by employing an object detection model similar to that of Hossen et al. [31].

hCaptcha Solver. hCaptcha is an image-based CAPTCHA service similar to reCAPTCHA. However, hCaptcha presents users with more realistic and even AI-generated images. We address two common variants of hCaptcha:

- **Variant 1 (Binary Selection).** Object-Identifying hCaptcha is similar to reCAPTCHA v2 (See Figure 4b). But it offers more complex challenge descriptions, enriched with extra context on styles or relations to other objects (e.g., someone playing football). Hence, we approach the problem as an open-set visual question answering (VQA) model. In a typical VQA task, a model is presented with an image alongside a text-based question about the visual content. The model then generates an answer, which can be a simple "yes" or "no" or a more complex textual answer. Given a CAPTCHA challenge specifying the object description as x, we transform it into a question "Is this a/an x?" for each candidate grid, then output an answer of either "yes" or "no". As a result, the solver selects all the grids with "yes".

- **Variant 2 (Area Selection).** This is an emerging variant (See Figure 4c) that asks the user to point out a specific location within an image. To solve these more advanced challenges, we employ the off-the-shelf tool *hCaptcha Challenger* [6].

For reCAPTCHA v2 and hCaptcha, if the deep-learning-based solver fails, we defer the challenge to a visual language model (VLM) agent and ask for a controlled response (e.g., click on squares 1, 5, 7 or coordinates (x, y)).

Slider CAPTCHA Solver. Slider-based CAPTCHAs require users to slide a puzzle piece into an empty spot on a background image [54, 67]. In addition to the accuracy of the placement, the CAPTCHAs also analyze the sliding trajectory to detect automated behavior. For example, human users are unlikely to maintain a constant speed throughout the slide.

We design our slider solver with traditional computer vision techniques. First, we identify the the background image and puzzle piece elements from the webpage source code. We then apply pre-processing techniques like Gaussian blurring for denoising and grayscale conversion followed by Sobel edge detection for edge sharpening. Next, we use the puzzle piece as a template for template matching, locating a similar region within the background image. Finally, the solver uses easing functions to simulate a human-like dragging trajectory.

Rotation CAPTCHA Solver. Rotation CAPTCHAs require users to adjust randomly rotated images to their upright orientation [29]. These challenge images usually feature natural and man-made landscapes.

In this work, we treat the image rotation problem as a regression task to predict the current degree of rotation for the challenge image. Once the rotation angle is determined, the solver can interact with the CAPTCHA to correct the orientation. To construct our model, we adopt EfficientNet [60], pretrained on ImageNet. We further fine-tune the model using randomly rotated samples from the Landscape Dataset [7], a community-contributed collection of 7,268 images that depict natural and man-made landscapes. Cosine distance to the ground-truth angle serves as the training loss.

4.4 Adversarial Countermeasure

Since PhishDecloaker orchestrates several deep-learning models, it may be vulnerable to adversarial attacks at runtime. We identify two plausible attack scenarios, i.e., system-level attack and model-dependant attack. A system-level attack introduces blurring, noise, or other obfuscations to CAPTCHA challenges, hindering the models' ability to identify content. This commonly occurs when CAPTCHAs detect suspicious activities from an IP address and present more challenging images. This attack is model-agnostic and does not target any specific model. A model-dependent attack exploits existing gradient-based methods [28, 38, 40, 44, 51] to deliberately perturb inputs and induce incorrect predictions. Both our CAPTCHA detector and recognition models could be susceptible to this type of attack.

To counter the former, we use adversarial training for all our deep-learning models. This approach aims to enhance model robustness by introducing adversarial examples during training. These examples are created by applying random augmentations to the original input data, potentially leading the model to make incorrect predictions. In our work, we consider the following types of augmentations such as Random Mask, Gaussian Noise, and Gaussian Blur. We mix adversarial and clean samples at a ratio of 6:4. To counter the latter, we implement PhishDecloaker with the gradient masking technique as proposed in [35]. Specifically, we replace the ReLU activation function with a step ReLU function defined as $f(x) = \max(0, \alpha \cdot \lceil \frac{x}{\alpha} \rceil)$, where α is the discretization parameter. This renders the activations non-differentiable, effectively zeroing out the gradients.

5 Experiments

We evaluate PhishDecloaker with the following questions:

- **RQ1 (Effectiveness):** What is the performance of PhishDecloaker in revealing CAPTCHA-cloaked phishing kits?
- **RQ2 (CAPTCHA Detection & Recognition):** What are the performances of PhishDecloaker's CAPTCHA detection and recognition components?
- **RQ3 (CAPTCHA Solving):** What is the performance of PhishDecloaker's CAPTCHA solvers?
- **RQ4 (Ablation Study):** What are the alternatives for PhishDecloaker's design, and how do they perform?
- **RQ5 (Adversarial Attacks):** Is PhishDecloaker robust against adversarial attacks on its deep learning models?
- **RQ6 (Field Study):** Can PhishDecloaker help discover more zero-day phishing websites in the wild?

To address each question, we first introduce the experimental settings (e.g., model training), followed by the objective metrics used to evaluate each research question. More experiment details are available at [9].

5.1 RQ1: Experimental Effectiveness

5.1.1 Experiment Setup

Phishing Detectors. We select Phishpedia [35] and PhishIntention [36] for their state-of-the-art performance on detecting zero-day phishing websites. Following the instructions of both detectors, we use a reference list of 277 phishing targets (i.e., Facebook, Bank of America, etc).

Table 2: Phishing detection rate on DynaPD. The percentages are calculated as changes relative to the baseline (No Cloaking). The average runtime overhead is computed for each module (CAPTCHA detection, CAPTCHA recognition and CAPTCHA solver) and concatenated by "+".

Group	No Cloaking	After Cloaking			
		reCAPTCHA v2	hCaptcha	GeeTest Slide	Rotation
Phishpedia	0.73	0.00 (↓100%)	0.00 (↓100%)	0.00 (↓100%)	0.00 (↓100%)
PhishIntention	0.53	0.00 (↓100%)	0.00 (↓100%)	0.00 (↓100%)	0.00 (↓100%)
Phishpedia + PhishDecloaker	0.73	0.57 (↓22.2%)	0.29 (↓59.8%)	0.69 (↓5.1%)	0.61 (↓16.1%)
PhishIntention + PhishDecloaker	0.53	0.41 (↓22.3%)	0.21 (↓59.8%)	0.50 (↓5.1%)	0.45 (↓16.0%)
Runtime Overhead (s)	-	0.13 + 0.05 + 44.17	0.12 + 0.05 + 8.81	0.13 + 0.06 + 5.12	0.13 + 0.08 + 5.01

Phishing Dataset. We apply our hardening framework, Cloaken, to the DynaPD dataset [37]. The DynaPD dataset comprises approximately 6k deployable and interactable phishing kits, providing a replicable environment to study CAPTCHA cloaking on phishing kits. Due to limitations in the reference lists of phishing detectors [35,36], we filter out phishing kits targeting sites not included in the reference list. This filtering results in a dataset of 2,960 phishing kits for our study. Cloaken cloaks each phishing kit with 4 CAPTCHA instances under the category of reCAPTCHA, hCaptcha, slider, and rotation, none of these CAPTCHA instances are used for training the models.

Measurement. We evaluate whether PhishDecloaker can help the phishing detectors to recover its access to the phishing content. Specifically, we evaluate detection rate of a phishing detector on DynaPD, r_1, its detection rate on the different types of cloaked phishing website variants, r_2, and its detection rate after equipped with PhishDecloaker, r_3. Note that, PhishDecloaker is not designed for improving the precision of existing phishing detectors, thus we only evaluate the recall measurement in the study.

Environment. We use Chrome version 114 and ensure a clean browser state for each session, i.e., with no caches or cookies preserved between consecutive requests. To conceal any indications of a headless browser and automation, we modify the requests and web browser characteristics, such as customizing User-Agent headers and adjusting to the Navigator object properties, as well as modifying to WebGL vendor. All solvers operate from a single IP address and machine with 20 CPU cores, 125G memory, and A100 GPU.

5.1.2 Results

Table 2 shows the overall experiment results. On the 2.9k phishing websites without any cloaking, Phishpedia and PhishIntention achieve detection rates of 72.6% and 53.0% respectively. Any CAPTCHA-cloaked variants can compromise their effectiveness, dropping the detection rate to 0%. In contrast, PhishDecloaker can recover their detection rates back to a percentage of their original performance: 78% on reCAPTCHA, 40% on hCaptcha, 95% on slider CAPTCHA, and 84% on rotation CAPTCHA. Overall, PhishDecloaker's contribution comes with acceptable runtime cost. Note that the main overhead lies in CAPTCHA solving. Specifically, solving a reCAPTCHA instance takes an average of 44.17s. This is because reCAPTCHA can iteratively replace selected images with new ones and prompt the user to click "verify" once there are no target objects (e.g., motorcycle) left. This iterative interaction incurs a significant runtime overhead.

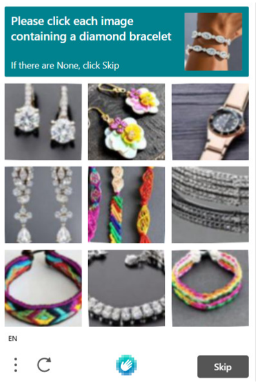
5.1.3 Qualitative Analysis

Next, we further investigate and categorize the CAPTCHAs which PhishDecloaker cannot address. The reason lies in as follows.

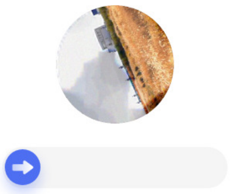
Incapability of the off-the-shelf solvers. We observe that the off-the-shelf solvers (e.g., hCaptcha solver) can have their limitations. Figure 5a shows an example where PhishDecloaker successfully detect and recognize the hCaptcha type but the hCaptcha solver fails to solve the challenge of *click each image containing a diamond bracelet*. Our investigation shows that the model mistakenly recognize earrings as bracelet, which are of similar visual semantics. A potential remedy is to retrain the model to further distinguish the embedding space of the model. We will discuss on how to improve the capability in Section 6.

Human verification beyond CAPTCHA. We discover that some CAPTCHAs such as reCAPTCHA v2 verifies a human using more than an interactive challenge. It may include mouse behavior analysis and browser fingerprinting, which will block our visits despite solving the challenge. We acknowledge the significance of these methods. Since there are relevant works [31,64] on this direction (see Appendix A.3), we limit our focus on countering CAPTCHA cloaking, which complements these works. A hybrid security crawler with various decloaking techniques is vital for effectiveness.

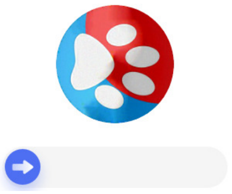
The restriction of training dataset. Finally, we find that our customized solvers might be limited by our training dataset. In general, our regression model (see Section 4.3) learns upright orientation for different pictures. If a picture to be rotated is deviated from the training dataset, the model might fail to predict its rotation degree effectively. For example, Figure 5b


Please click each image containing a diamond bracelet
If there are None, click Skip
EN
Skip

(a) an instance of hCaptcha (V1) which cannot be solved by PhishDecloaker.



(b) a solved rotation CAPTCHA instance.



(c) an unsolved rotation CAPTCHA instance.

Figure 5: Example cases in solving CAPTCHAs

and Figure 5c manifest different genres of images, which leads to performance disparity. Although we do not claim contribution on solving any particular CAPTCHA instance, we will discuss potentially better solutions in Section 6.

5.2 RQ2: CAPTCHA Detection & Recognition

5.2.1 CAPTCHA Detection

Dataset Collection. To collect the training dataset of detecting CAPTCHA, we adopt XDriver [23] to crawl the websites listed in the Alexa top 1-million websites. It automatically locates the forms on the page, fills in all form inputs with simulated data, and submits the form in order to trigger CAPTCHAs. It then captures screenshots of these pages, which we manually annotate to identify the bounding boxes for any CAPTCHAs present. Due to ethical and security considerations, we strictly limit our crawling to a single instance per website, with a maximum depth of 2. Over two weeks, we collected and labeled 1,764 webpage screenshots containing CAPTCHAs. We employ data augmentation to enrich our dataset with additional synthetic samples, bringing the total to 10,680 webpage screenshots. Examples of the synthetic samples generated are shown in Appendix A.1. We perform a 9:1 train-test split, where 9,612 samples are for training and 1,068 samples for testing.
Training Settings. We use the training framework provided by the authors of [33], which is built upon OpenMMLab Detection Toolbox [8]. The OLN object detection model uses Faster-RCNN pre-trained on ImageNet as its backbone, with the RPN and RoI head modified as described in Section 4.1. We train the model for 8 epochs, with a batch size of 2 per GPU, using Stochastic Gradient Descent with a learning rate

Table 3: Performance for CAPTCHA Recognition on Open-set CAPTCHAs.

Class	Precision	Recall	F1-Score
arkose_select	0.93	0.91	0.92
capycaptcha_drag	0.88	0.58	0.70
dicecaptcha_qa	0.97	0.68	0.80
funcaptcha_select_1	0.99	0.87	0.93
funcaptcha_select_2	0.98	0.48	0.64
funcaptcha_select_3	0.88	0.52	0.65
funcaptcha_select_4	0.62	0.91	0.74
funcaptcha_select_5	1.00	0.53	0.69
funcaptcha_select_6	0.88	0.72	0.79
keycaptcha_drag	0.93	0.75	0.83
mtcaptcha_text	0.46	0.63	0.53
Average	**0.86**	**0.69**	**0.75**

of 0.02 and momentum of 0.9.
Measurement. We use the mean average precision (mAP) and mean average recall (mAR) to evaluate the performance, which are the standard metrics for evaluating the completeness and redundancy of the reported objects in object detection tasks [33, 53]. The mAP and mAR are computed over IoU thresholds ranging from 0.5 to 0.95.
Results. The OLN detector achieves a mean average precision and recall of 0.92 and 0.97 respectively.

5.2.2 CAPTCHA Recognition

Dataset Collection. We have collected a total of 6,612 CAPTCHA samples spanning 38 classes, sourced from demo websites (e.g., NetEase, Tencent, Arkose Labs), official API keys provided by vendors (e.g., Google, hCaptcha, GeeTest), and open-source community datasets. During training, we employed a 9:1 train-test split, allocating 5,950 samples for training and 662 samples for testing. This dataset serves as the template database at deployment time.
Training Settings. As for the feature extractor, the visual branch employs a ResNet-50 model pre-trained on ImageNet with its classification head removed. This branch takes a resized CAPTCHA region of dimensions 224×224 as input and outputs the visual embedding. The textual branch is adapted from EasyOCR [5]. It utilizes a Character-Region Awareness for Text (CRAFT) model [18] pre-trained on SynthText for bounding box detection and a Convolutional Long Short Term Memory (CLSTM) [55] model pre-trained with the STR framework [17] for textual embedding projection. During training, we freeze the textual branch and fine-tune all other branches using Sub-center ArcFace [21] as described in Section 4.2. We train the model for 100 epochs, with a batch size of 2 per GPU, using Stochastic Gradient Descent with a learning rate of 0.02 and a momentum of 0.9.

Measurement. We evaluate the training and testing accuracy of our model in recognizing a particular CAPTCHA type. In addition, we evaluate the generalizability of our model by whether our model can recognize the new CAPTCHA types without retraining the model. We argue that utilizing deep Siamese learning for CAPTCHA recognition can improve performance in open-set scenarios. To validate this, we first collect an additional 11 CAPTCHA types not present in the training datasets of our CAPTCHA detection and recognition models. These newly acquired CAPTCHAs are superimposed onto random webpage screenshots to create open-set test samples. We then update PhishDecloaker's template database with references from these new CAPTCHA classes and assess the system's ability to correctly classify these novel CAPTCHAs.
Results. Table 3 presents our results on the open-set dataset. Our system yields satisfactory performance in correctly identifying unseen CAPTCHA types, with an average precision of 86.0% and an average recall of 69%.

5.3 RQ3: CAPTCHA Solving

CAPTCHA Benchmark. Our CAPTCHA benchmarking dataset includes reCAPTCHA v2, hCaptcha (V1&2), slider CAPTCHA, and rotation CAPTCHA. For reCAPTCHA v2, hCaptcha (V1&2), we test different difficulty levels: easy, moderate, and difficult, which is categorized by the CAPTCHA vendor. As for slider CAPTCHA, we include three different versions from GeeTest, Tencent, and NetEase. Lastly, we select Baidu's rotation CAPTCHA for evaluation. In general, we generate 200 CAPTCHA variants for each version. In total, we evaluated 10 CAPTCHA versions in the study.
Measurement. For reCAPTCHA v2, hCaptcha, all slider CAPTCHA variations, and Baidu rotation CAPTCHA, we determine their success rate by calculating the proportion of successfully resolved CAPTCHA sessions compared to the total number of requested CAPTCHA sessions. To elaborate, during each CAPTCHA session, the solver may encounter one or more consecutive CAPTCHA challenges. In order for a CAPTCHA session to be considered as successfully solved, the solver must successfully complete all presented challenges and receive a confirmation of success (e.g., a green checkmark in reCAPTCHA v2).

Additionally, to further analyze our rotation CAPTCHA solver, we evaluate it on the test set in the Landscape Dataset. The test samples are randomly rotated between 0 to 360 degrees. We use mean angular error (MAE) as the evaluation metric. Given a batch of N predicted angles θ_i and their ground truth (rotated) angles $\hat{\theta}_i$, MAE is defined as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\Big(180 - \big||\theta_i - \hat{\theta}_i| - 180\big|\Big) \qquad (2)$$

This metric quantifies the average angular discrepancy between the predicted and actual angles.

Table 4: CAPTCHA Metrics and Detection Rates

CAPTCHA	Category	Solving rate
reCAPTCHA v2	Easy	75.5%
	Moderate	74.0%
	Difficult	35.0%
hCaptcha	Easy	85.0%
	Moderate	92.0%
	Difficult	71.0%
Slider	GeeTest	95.0%
	Tencent	89.0%
	NetEase	100.0%
Rotation	Baidu	74.5%

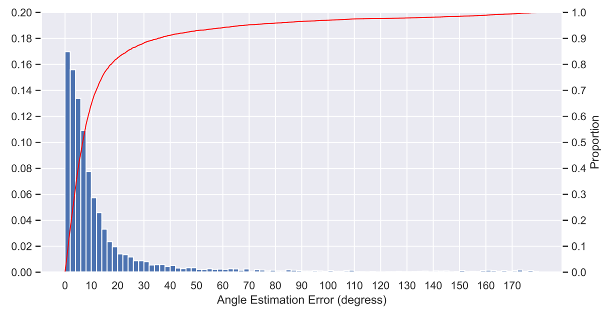


Figure 6: Histogram of angular errors and their cumulative distribution (red line) for the regression model on the test set in Landscape Dataset.

Results. For reCAPTCHA and hCaptcha, we observe that our solvers perform well on easy and moderate challenges. Performance degrades for difficult cases, with reasons discussed in Section 5.1. Using the template matching algorithm, the slider CAPTCHA solver performs uniformly well across three different service providers, because that the algorithm is robust against noise and distortion in the background images.

The rotation solver achieves a mean angular error of 15.62. Figure 6 displays a histogram of the angular errors along with its cumulative distribution. We observe that more than 90% of the test samples exhibit an angular error of less than 35 degrees, and the histogram follows a long-tail distribution. This indicates that the solver was able to reorient the majority of images. We then conducted a test using Baidu's rotation CAPTCHA service, limiting the number of attempts to 200. The solver achieves a solving rate of 74.5%.

5.4 RQ4: Ablation Study

We explore alternatives for PhishDecloaker's architecture design on recognizing a CAPTCHA type from a webpage screenshot (i.e., a pipelined task of CAPTCHA detection and recognition). We test the following alternatives:

Table 5: Results of our abalation study.

Configuration		Result	
Detector	Recognizer	Precision	Recall
F-RCNN	ResNet-50	0.89	0.84
F-RCNN	Single-branch Siamese	0.92	0.84
F-RCNN	Dual-branch Siamese	0.91	0.82
OLN	ResNet-50	0.85	0.77
OLN	Single-branch Siamese	0.92	0.86
OLN	Dual-branch Siamese	**0.93**	**0.86**

- **OP1 (alternative recognizer):** How does a deep classifier (implemented by ResNet-50) compare with Siamese metric learning model in recognizing CAPTCHA types?
- **OP2 (alternative object detector):** How does Faster R-CNN compare with OLN in detecting CAPTCHAs?
- **OP3 (alternative branch design):** How does a dual-branch Siamese (i.e., visual & OCR-aided textual branch) compare with a single-branch Siamese (i.e., visual branch)?

Dataset. To evaluate the precision and recall of CAPTCHA recognition, we overlay the test samples from our CAPTCHA recognition dataset (Section 5.2.2) onto 622 screenshots of benign webpages and supplement them with 100 webpage screenshots without CAPTCHAs.
Results. Table 5 shows our results. We observe that PhishDecloaker's design is the most optimal. Compared to Siamese models, a classifier achieves less accuracy, especially in those screenshots without CAPTCHAs. Compared to Faster R-CNN, OLN manifests better recall. Finally, the dual-branch Siamese design has a marginal improvement over single-branch design by considering textual features (0.93 over 0.92 in precision). Security practitioners can make a trade-off between model complexity and performance.

5.5 RQ5: Robustness Against Adversaries

In this section, we assess PhishDecloaker's robustness to evasion attacks by generating custom CAPTCHA images that target specific components of PhishDecloaker. We then evaluate the system's overall resilience.

5.5.1 Adversaries

We model our adversary as a phisher with no constraints in time and computing resources to deploy evasion attacks. The adversary's aim is to create a CAPTCHA-cloaking page that remains undetected by PhishDecloaker. To this end, we conduct adversarial attacks on detection, recognition, and the solving components, with the assumption that the adversary possesses *perfect* knowledge of PhishDecloaker's design.

Attacks on CAPTCHA Detection. We conduct DPatch attack [38] to compromise the CAPTCHA detection component of PhishDecloaker. DPatch generates adversarial patches that can be applied to a webpage. In our case, the patches are untargeted, To create untargeted patches, DPatch finds a patch pattern $\vec{P}_u$ that maximize the loss of the object detector to the true class label $\vec{y}$ and bounding box label $\vec{B}$ when the patch is applied to a webpage screenshot x using "apply" function A, as shown in Equation 3 [38]. The apply function $A(x,P)$ means adding patch P onto webpage screenshot x. As a result, an object detection model can potentially fail to locate the correct region containing CAPTCHAs.

$$\vec{P}_u = \arg\max_P \mathbb{E}_x \left[L(A(x,P);\vec{y},\vec{B}) \right] \tag{3}$$

Attacks on CAPTCHA Recognition. We conduct adversarial attacks on CAPTCHA images including Fast Gradient Sign Method (FGSM) [28], Jacobian Saliency Map Attack (JSMA) [51], Projected Gradient Descent (PGD) [40], and DeepFool [44] to compromise the CAPTCHA recognition component of PhishDecloaker. In the attack, we assume that the attacker can access the white-box model but cannot poison or modify PhishDecloaker deployed online,
Augmentation Attacks. Additionally, we conduct generic image augmentation attacks on PhishDecloaker by performing 6 types of transformations on the CAPTCHA image and overlay the transformed CAPTCHA onto random webpage screenshots. Specifically, they are Random Stretch, Gaussian Noise, Random Crop, Random Mask, Salt & Pepper, and Gaussian Blur. Appendix A.2 visualizes the attacks. Those augmentations are commonly found in phishing webpages [16]. Different from generating the adversarial images for CAPTCHA detection and recognition component, those adversarial samples are visible.
Dataset. We apply all the above attacks on the dataset described in Section 5.2.2.
Measurement. We measure the performance of PhishDecloaker before and after attacks in terms of the accuracy of the results of a pipelined CAPTCHA detection and CAPTCHA recognition. We evaluate the robustness against the adversaries by the perturbation of overall CAPTCHA recognition rate before and after the attack. We do not evaluate the solving accuracy as we cannot change the online CAPTCHAs.

5.5.2 Results

Table 6 and Table 7 show that the accuracy loss under attacks. We can see that our adopted gradient masking technique is effective in defending the state-of-the-art gradient-based adversarial attack on the deep learning models, i.e., CAPTCHA detection and recognition models.

Furthermore, Table 8 shows the performance of PhishDecloaker in recognizing augmented CAPTCHAs, including attacks such as Random Stretch and Gaussian Noise, among

Table 6: The robustness of CAPTCHA recognition model against diverse adversarial attack.

Attack	Accuracy (no Def.)	Accuracy (with Def.)
No Attack	0.97	1.00
JSMA	0.50 (-48.5%)	1.00 (-0.0%)
PGD	0.12 (-87.6%)	1.00 (-0.0%)
DeepFool	0.07 (-92.8%)	1.00 (-0.0%)
FGSM	0.06 (-93.8%)	1.00 (-0.0%)

Table 7: The robustness of CAPTCHA detection model against adversarial attack.

Attack	mAP (no Def.)	mAP (with Def.)
No Attack	97.70	91.55
DPatch	54.65 (-44.1%)	85.71 (-6.4%)

others. We observe that transformation-based attacks can be effectively countered through adversarial training. Although the adversarial training process has only a subset of transformations (i.e., Random Mask, Gaussian Noise and Blur), the model performed well against other attacks, suggesting that learned transformations can be generalized to effectively handle unseen transformations.

Table 8: The robustness of PhishDecloaker against augmentation attacks.

Attack	Accuracy (no Def.)	Accuracy (with Def.)
No Attack	0.97	1.00
Random Stretch	0.95 (-1.9%)	0.96 (-4.0%)
Gaussian Noise	0.87 (-10.2%)	0.94 (-6.0%)
Random Crop	0.82 (-15.3%)	0.83 (-17.0%)
Random Mask	0.76 (-21.5%)	0.90 (-10.0%)
Salt and Pepper	0.33 (-66.4%)	0.92 (-8.0%)
Gaussian Blur	0.18 (-82.0%)	0.93 (-7.0%)

5.6 RQ6: Field Study

We further design a field study to evaluate the emergence of CAPTCHA-cloaked phishing websites in the real world.

5.6.1 Experiment Setup

URL Source. We crawl fresh URLs from Certstream [3] in real-time for 4 weeks, which provides domains with newly issued TLS/SSL certificates.

Study Groups. We prepare 6 different study groups to analyze crawled sites for phishing, each using PhishIntention as the base phishing detector for its state-of-the-art performance in detecting zero-day phishing websites. Group 1 is a *control* group with no JavaScript (JS) rendering or decloaking techniques. Group 2 has only JS rendering. Groups 3 to 6 have JS rendering and each is equipped with a type of decloaking technique below:

- **Anti-interaction-cloaking (AI):** Automatically closes alert, permission, and notification windows. It also randomly moves and clicks the mouse after the page is loaded.
- **Anti-fingerprint-cloaking (AF):** Randomizes its user agent and cookie storage, spoofs its referrer, and uses a stealth headless browser.
- **Anti-behavior-cloaking (AB):** Automatically follows all redirects and waits for 5 seconds if the page is blank (i.e., loading). It retries up to 3 times if the page fails to load.
- **Anti-CAPTCHA-cloaking (AC):** Uses PhishDecloaker to automatically detect and solve CAPTCHAs.

Validation & Monitoring. We manually inspect and confirm reported phishing websites. Further, we submit the confirmed phishing websites to VT and track their lifespan. We consider a manually confirmed phishing website as zero-day if it is not reported by any of the detectors in VT. In addition, we measure the time taken for sites to expire (*time-to-takedown*) and blacklisted by VT (*time-to-blacklist*).

5.6.2 Results

In this field study, we captured totally 869 unique phishing websites by all 6 study groups. Of these, 7.6% were CAPTCHA-cloaked phishing websites, all of which were discovered solely by PhishDecloaker. Table 9 further details how each decloaking groups contribute to phishing discovery. Group 3 (e.g., random mouse movement), Group 5 (e.g., repetitive visits), and Group 6 (with PhishDecloaker) report unique phishing websites, i.e., the phishing websites reported only by the specific group. We consider Group 3-5 as basic and traditional decloaking functions (e.g., randomly moving the mouse and repetitive visits to a website), which still play an important role in report zero-day phishing websites (i.e., Group 6 reports 203; Group 5 reports 198; and Group 3 reports 197). Nevertheless, compared to those traditional decloaking groups, CAPTCHA-based cloaking is emerging, which ranks the second in terms of unique ratio and ranks the first in terms of the number of discovered zero-day phishing websites, which raises alarm to the phishing detection community. Further, the result indicates a practical or commercial phishing decloaking shall be *hybrid* to handle different cloaking techniques.

Next, we investigate the features of CAPTCHA-cloaked phishing websites regarding the sectors of their target brands, CAPTCHA types, lifespan, and time to be blacklisted.

Table 9: Field study results of each decloaking group on Certstream URLs. PI: PhishIntention. JS: JavaScript rendering. AI: anti-interaction-cloaking. AF: anti-fingerprint-cloaking. AB: anti-behavior-cloaking. AC: anti-CAPTCHA-cloaking.

Group	Setup	Unique Ratio	# 0-Days	# Phishing
G1	PI	0.0%	101 (−0.0%)	361 (−0.0%)
G2	PI + JS	0.0%	176 (↑74.3%)	582 (↑61.2%)
G3	PI + JS + AI	14.1%	197 (↑95.0%)	710 (↑96.7%)
G4	PI + JS + AF	0.0%	165 (↑63.4%)	543 (↑50.4%)
G5	PI + JS + AB	7.4%	198 (↑96.0%)	692 (↑91.7%)
G6	PI + JS + AC	10.2%	203 (↑101.0%)	648 (↑79.5%)

Table 10: Top-5 targeted sectors by ordinary and CAPTCHA-cloaked phishing sites.

Ordinary	%	CAPTCHA-Cloaked	%
Telecommunications	23.8	Cryptocurrency	43.9
Social Networking	22.8	Social Networking	19.3
Gambling	12.5	Logistics / Courier	15.8
Online Services / Software	12.3	Government Services	8.8
Financial / Insurance	10.1	Financial / Insurance	5.3

Target Sectors. We further investigate the sectors of the target brands of traditional phishing websites and CAPTCHA-cloaked phishing websites as showed in Table 10. We observe that CAPTCHA-cloaked phishing websites are more likely to target vibrant sectors like cryptocurrency (e.g., Coinbase and Trust Wallet). In contrast, traditional phishing websites still target sectors such as telecommunication (e.g., Orange and AT&T).

CAPTCHA Types and Their Usage. As showed in Figure 7, phishers tend to use *free* and *convenient* CAPTCHA services. The distribution of CAPTCHA types used by benign websites are: reCAPTCHA v2 (76.0%), hCaptcha (19.3%), Text CAPTCHA (4.0%), Press & Hold (0.6%), Slider (0.1%), whereas it is mainly hCaptcha (77.3%) and reCAPTCHA v2 (22.7%) on phishing sites. Interestingly, we find that fewer than 20% of CAPTCHA API keys (i.e., the keys used to access commercial CAPTCHA services) account for more than 55% of the discovered CAPTCHA-cloaked phishing sites. One hCaptcha key was even found to be shared across 19 sites. It potentially indicates that CAPTCHA-cloaked phishing attackers might also be sensitive to the "attacking cost".

Life Span and Time to Blacklist. Different from our expectation, CAPTCHA-cloaked phishing websites have shorter lifespan compared to ordinary phishing sites (9.7 vs. 13.2 hours), as showed in Figure 8. However, it takes blacklist-based detectors such as Google Safe Browser and SmartScreen more time to put the URLs of CAPTCHA-cloaked phishing sites in their blacklist, as showed in Figure 9. CAPTCHA-cloaked

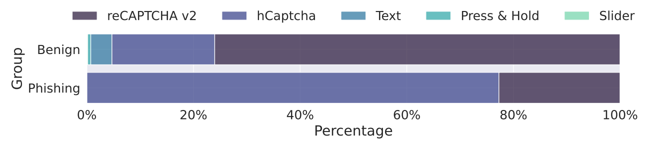


Figure 7: Distribution of CAPTCHA types on benign and phishing sites.

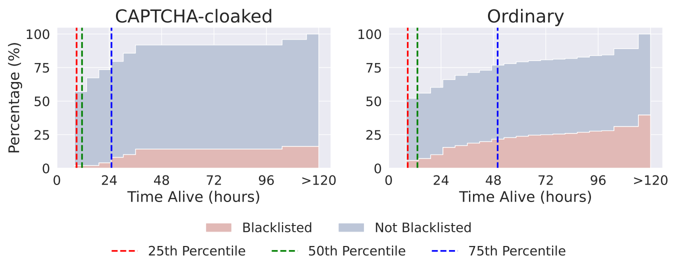


Figure 8: Cumulative distribution of life span for CAPTCHA-cloaked and ordinary phishing sites.

phishing sites take a median time of 16 hours to be blacklisted, which is 45.5% longer than ordinary phishing sites (11 hours). The result indicates that CAPTCHA-cloaked phishing websites are more active and evasive for traditional phishing detectors.

Overhead. In this study, the median time of PhishDecloaker on CAPTCHA detection and recognition are 0.4s and 0.3s respectively, making Group 6 53.1% (0.68s) slower than Group 2. Although median time for CAPTCHA solving is 15.3s, it can be decoupled and processed asynchronously.

6 Discussion

Limitations. Although PhishDecloaker can be extended to include new CAPTCHA types for CAPTCHA detection and recognition, it is limited by the number of supported CAPTCHA solvers. When PhishDecloaker encounters a rare CAPTCHA without a corresponding solver in its repository, we offer two suggestions. First, PhishDecloaker can integrate with existing CAPTCHA solving services (e.g., 2Captcha) that rely on paid labour. These services provide a API endpoint for each different CAPTCHA type. In this case, PhishDecloaker can identify the CAPTCHA type and call the right API for the task. Second, PhishDecloaker can automatically notify in-house human operators of unsolvable CAPTCHAs. Nevertheless, in the era of Artificial General Intelligence (AGI), we expect that emerging AI solutions can further empower PhishDecloaker to achieve better performance.

Ethical Considerations. Our empirical study submits URLs of self-hosted phishing kits to anti-phishing entities, which may have a negative impact to the community. More specifically, security crawlers may spend unnecessary resources on

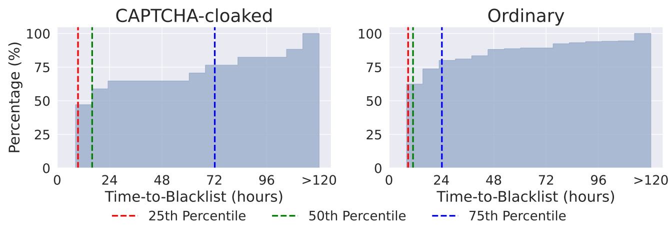


Figure 9: Cumulative distribution of time taken to be blacklisted by SmartScreen or GSB for CAPTCHA-cloaked and ordinary phishing sites.

analyzing simulated threats and innocent visitors may stumble upon these sites. To mitigate this, we strictly follow the established guidelines of previous works [13, 46, 47] by randomly generating long URLs, submitting them only to anti-phishing entities through proper channels, restricting the total number of submissions to 500 per entity, and use defanged phishing kits that do not store nor share credentials. Besides, PhishDecloaker can be exploited to compromise benign websites. We opt to keep it closed-source, sharing exclusively with trusted researchers, security firms, and government agencies. We can also deploy it as a restricted cloud service, where access key is revoked if abused.

Future Work. Recently, vision-language foundation models have demonstrated extraordinary emergent abilities on web navigation tasks [22, 25]. These models enable transformative generalization and are capable of solving wide ranges of interactive decision making problems in the wild [45]. Hence, it is possible to study the feasibility of these models as zero-shot or few-shot generalized CAPTCHA solvers.

7 Related Work

Phishing Detection. Conventional phishing detection systems such as SmartScreen, Google Safe Browsing, and OpenPhish rely on blacklists, which are updated through user reports, automatic crawling, and manual verification. However, this method is limited by delays in list updates and frequently misses short-lived phishing campaigns [49].

To automate verification, feature-engineering-based solutions [20, 24, 27, 34, 39, 61, 63] use feature extraction and classification techniques, focusing on HTML code, URLs, domains, and screenshots. Despite their utility, these solutions are inflexible and susceptible to code obfuscation, resulting in rapid data obsolescence. To overcome these limitations, reference-based solutions [12, 15, 35–37, 42] employ deep-vision techniques to compare the representations of a phishing page against a pre-defined reference list, determining its target brand. These approaches are both extensible and explainable, advancing the state-of-the-art in phishing detection.

Cloaking. Phishing websites use advanced cloaking techniques to evade detection [14, 41, 48, 59, 64, 65]. Two main types exist: server-side and client-side cloaking. Server-side cloaking identifies users via HTTP requests, often using .htaccess or PHP scripts [19, 41, 46, 47, 49, 65]. It employs IP and keyword blacklists, geolocation, and user-agents to filter traffic. Countermeasures include multiple visits with spoofed IPs and user-agents [13, 32, 35, 36, 64].

Client-side cloaking operates within browsers. It employs browser fingerprinting and user interaction, such as pop-ups or CAPTCHAs [13, 41, 59, 64]. It also manipulates bot behavior to delay loading times [64]. Despite its rising popularity, client-side cloaking challenges anti-phishing engines [41, 47]. Though lacking a systematic approach, some advances have been made. For example, Crawlphish uses JavaScript force execution to detect client-side cloaking but focuses more on post-hoc analysis than real-time detection [64].

CAPTCHA Solving. Deep learning models have been used to solve specific CAPTCHA types [26, 31, 57, 66, 67], but no system exists for automatically identifying arbitrary CAPTCHAs, consistent with [59]. Some ad hoc solvers include: Sivakorn et al. [57] tackled Google reCAPTCHA v2 by exploiting challenge instructions. They used image annotation services and Word2Vec to match tags with challenge text. Hossen et al. [31] employed a custom object detection model to recognize the objects present in the challenge images. For slider CAPTCHAs, Zhao et al. [67] developed an algorithm to match background and target images to identify the puzzle region, whereas Wu et al. [62] used object detection.

Unlike the above solutions, PhishDecloaker detects, recognizes, and then solves various CAPTCHAs using an expandable repository of CAPTCHA solvers for anti-phishing.

8 Conclusion

We studied CAPTCHA cloaking, a prevalent technique among phishing websites. Our empirical study showed that none of the selected phishing detectors could detect CAPTCHA-cloaked phishing. Motivated by this, we developed PhishDecloaker to automatically detect, recognize, and solve CAPTCHAs. Our experiment confirmed PhishDecloaker's ability to restore phishing detection rates of Phishpedia and PhishIntention on CAPTCHA-cloaked phishing kits, and it remained robust against adversarial attacks. We further confirmed its practical impact through a field study on Certstream URLs, revealing interesting behavior of 0-day CAPTCHA-cloaked phishing sites in the wild.

Acknowledgments

We thank Khurshid Juraev for his work on the early prototypes of reCAPTCHA v2 and hCaptcha solvers. This research is supported in part by National Natural Science Foundation of China (62172099, 23Z990203011), the Minister of Education, Singapore (MOET32020-0004), the National Research Foundation, Singapore, and Cyber Security Agency of Singapore under its National Cybersecurity Research and Development Programme (Award No. NRF-NCR_TAU_2021-0002) and A*STAR, CISCO Systems (USA) Pte. Ltd and National University of Singapore under its Cisco-NUS Accelerated Digital Economy Corporate Laboratory (Award I21001E0002), National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN), DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-008).

References

[1] AWPG. Phishing Activity Trends Report, 4th Quarter 2022. https://docs.apwg.org/reports/apwg_trends_report_q3_2022.pdf.

[2] BuiltWith®. CAPTCHA Usage Distribution in the Top 1 Million Sites. https://trends.builtwith.com/widgets/captcha.

[3] Certstream. https://certstream.calidog.io/.

[4] curl-impersonate. https://github.com/lwthiker/curl-impersonate.

[5] EasyOCR. https://github.com/JaidedAI/EasyOCR/tree/master.

[6] hCaptcha Challenger. https://github.com/QIN2DIM/hcaptcha-challenger.

[7] Landscape Dataset. https://github.com/koishi70/Landscape-Dataset.

[8] OpenMMLab Detection Toolbox and Benchmark. https://github.com/open-mmlab/mmdetection.

[9] PhishDecloaker's Website. https://sites.google.com/view/phishdecloaker/home.

[10] Trend Micro. Addressing CAPTCHA-Evading Phishing Threats With Behavior-Based AI Protection. https://www.trendmicro.com/vinfo/id/security/news/internet-of-things/addressing-captcha-evading-phishing-threats-with-behavior-based-ai-protection.

[11] Moataz AbdelKhalek and Ahmed Shosha. Jsdes: An automated de-obfuscation system for malicious javascript. In *proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–13, 2017.

[12] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. Visualphishnet: Zero-day phishing website detection by visual similarity. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1681–1698, 2020.

[13] 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.

[14] Bhupendra Acharya and Phani Vadrevu. A human in every ape: Delineating and evaluating the human analysis systems of anti-phishing entities. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 156–177. Springer, 2022.

[15] Sadia Afroz and Rachel Greenstadt. Phishzoo: Detecting phishing websites by looking at them. In *2011 IEEE fifth international conference on semantic computing*, pages 368–375. IEEE, 2011.

[16] Giovanni Apruzzese, Hyrum S Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Roundy. "real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 339–364. IEEE, 2023.

[17] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4715–4723, 2019.

[18] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9365–9374, 2019.

[19] 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.

[20] 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.

[21] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou. Sub-center arcface: Boosting face recognition by large-scale noisy web faces. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 741–757. Springer, 2020.

[22] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] Kostas Drakonakis, Sotiris Ioannidis, and Jason Polakis. The cookie hunter: Automated black-box auditing for web authentication and authorization flaws. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1953–1970, 2020.

[24] Birhanu Eshete, Adolfo Villafiorita, and Komminist Weldemariam. Binspect: Holistic analysis and detection of malicious web pages. In *Security and Privacy in Communication Networks: 8th International ICST Conference, SecureComm 2012, Padua, Italy, September 3-5, 2012. Revised Selected Papers 8*, pages 149–166. Springer, 2013.

[25] Hiroki Furuta, Ofir Nachum, Kuang-Huei Lee, Yutaka Matsuo, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*, 2023.

[26] Yipeng Gao, Haichang Gao, Sainan Luo, Yang Zi, Shudong Zhang, Wenjie Mao, Ping Wang, Yulong Shen, and Jeff Yan. Research on the security of visual reasoning {CAPTCHA}. In *30th USENIX security symposium (USENIX security 21)*, pages 3291–3308, 2021.

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

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

[29] R Gossweiler, M Kamvar, and S Baluja. A captcha based on image orientation. *Proc. ACM*, 2009.

[30] Dorjan Hitaj, Briland Hitaj, Sushil Jajodia, and Luigi V Mancini. Capture the bot: Using adversarial examples to improve captcha robustness to bot attacks. *IEEE Intelligent Systems*, 36(5):104–112, 2020.

[31] Md Imran Hossen, Yazhou Tu, Md Fazle Rabby, Md Nazmul Islam, Hui Cao, and Xiali Hei. An object detection based solver for {Google's} image {reCAPTCHA} v2. In *23rd international symposium on research in attacks, intrusions and defenses (RAID 2020)*, pages 269–284, 2020.

[32] 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 *2016 IEEE Symposium on Security and Privacy (SP)*, pages 743–758. IEEE, 2016.

[33] Dahun Kim, Tsung-Yi Lin, Anelia Angelova, In So Kweon, and Weicheng Kuo. Learning open-world object proposals without learning to classify. *IEEE Robotics and Automation Letters*, 7(2):5453–5460, 2022.

[34] 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.

[35] 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 (USENIX Security 21)*, pages 3793–3810, 2021.

[36] 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 22)*, pages 1633–1650, 2022.

[37] Ruofan Liu, Yun Lin, Yifan Zhang, Penn Han Lee, and Jin Song Dong. Knowledge expansion and counterfactual interaction for {Reference-Based} phishing detection. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4139–4156, 2023.

[38] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018.

[39] Christian Ludl, Sean McAllister, Engin Kirda, and Christopher Kruegel. On the effectiveness of techniques to detect phishing sites. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 4th International Conference, DIMVA 2007 Lucerne, Switzerland, July 12-13, 2007 Proceedings 4*, pages 20–39. Springer, 2007.

[40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[41] 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.

[42] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-similarity-based phishing detection. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, pages 1–6, 2008.

[43] Xianghang Mi, Xuan Feng, Xiaojing Liao, Baojun Liu, XiaoFeng Wang, Feng Qian, Zhou Li, Sumayah Alrwais, Limin Sun, and Ying Liu. Resident evil: Understanding residential ip proxy as a dark service. In *2019 IEEE symposium on security and privacy (SP)*, pages 1185–1201. IEEE, 2019.

[44] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[45] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[46] 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 *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1344–1361. IEEE, 2019.

[47] 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 (USENIX Security 20)*, pages 379–396, 2020.

[48] 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 Symposium on Electronic Crime Research (eCrime)*, pages 1–12. IEEE, 2018.

[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 ({USENIX} Security 20)*, 2020.

[50] Margarita Osadchy, Julio Hernandez-Castro, Stuart Gibson, Orr Dunkelman, and Daniel Pérez-Cabo. No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation. *IEEE Transactions on Information Forensics and Security*, 12(11):2640–2653, 2017.

[51] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[52] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proceedings of the Internet Measurement Conference*, pages 478–485, 2019.

[53] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[54] Andrew Searles, Yoshimichi Nakatsuka, Ercan Ozturk, Andrew Paverd, Gene Tsudik, and Ai Enkoji. An empirical study & evaluation of modern {CAPTCHAs}. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3081–3097, 2023.

[55] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[56] Chenghui Shi, Xiaogang Xu, Shouling Ji, Kai Bu, Jianhai Chen, Raheem Beyah, and Ting Wang. Adversarial captchas. *IEEE transactions on cybernetics*, 52(7):6095–6108, 2021.

[57] Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. I'm not a human: Breaking the google recaptcha. *Black Hat*, 14:1–12, 2016.

[58] Philippe Skolka, Cristian-Alexandru Staicu, and Michael Pradel. Anything to hide? studying minified and obfuscated code in the web. In *The world wide web conference*, pages 1735–1746, 2019.

[59] Karthika Subramani, William Melicher, Oleksii Starov, Phani Vadrevu, and Roberto Perdisci. Phishinpatterns: measuring elicited user interactions at scale on phishing websites. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 589–604, 2022.

[60] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[61] Rakesh Verma and Keith Dyer. On the character of phishing urls: Accurate and robust statistical learning classifiers. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 111–122, 2015.

[62] Danni Wu, Jing Qiu, Huiwu Huang, Lihua Yin, Zhaoquan Gu, and Zhihong Tian. Resnet-based slide puzzle

captcha automatic response system. In *Artificial Intelligence and Security: 6th International Conference, ICAIS 2020, Hohhot, China, July 17–20, 2020, Proceedings, Part III 6*, pages 140–153. Springer, 2020.

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

[64] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shawn Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1109–1124. IEEE, 2021.

[65] Penghui Zhang, Zhibo Sun, Sukwha Kyung, Hans Walter Behrens, Zion Leonahenahe Basque, Haehyun Cho, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, et al. I'm spartacus, no, i'm spartacus: Proactively protecting users from phishing by intentionally triggering cloaking behavior. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3165–3179, 2022.

[66] Yang Zhang, Haichang Gao, Ge Pei, Sainan Luo, Guoqin Chang, and Nuo Cheng. A survey of research on captcha designing and breaking techniques. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 75–84. IEEE, 2019.

[67] Binbin Zhao, Haiqin Weng, Shouling Ji, Jianhai Chen, Ting Wang, Qinming He, and Reheem Beyah. Towards evaluating the security of real-world deployed image captchas. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, pages 85–96, 2018.

A Appendix

A.1 Summary of Datasets

CAPTCHA Detection Dataset (Usage: see 5.2.1). Contents: 19,680 webpage screenshots (1920×1080), 10,680 with annotated CAPTCHA bounding boxes, 9,000 without.
CAPTCHA Recognition Dataset (Usage: see 5.2.2). Contents: 6,612 CAPTCHA images distributed across 38 classes.
CAPTCHA Open-set Dataset (Usage: see 5.2.2). Contents: 1,500 webpage screenshots (1920×1080), all of which have annotated CAPTCHA classes spanning 15 different categories.
Ablation Dataset (Usage: see 5.4). Contents: 722 webpage screenshots (1920×1080), 622 with CAPTCHAs spanning 38 classes, 100 without.
Landscape Dataset (Usage: see 4.3). Contents: 7,268 natural and man-made landscape images (320×180).

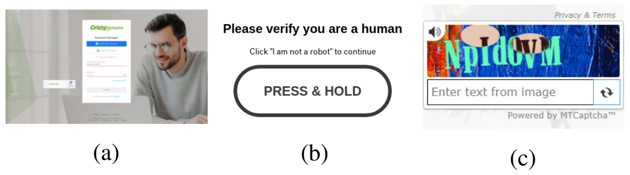


Figure 10: Examples of CAPTCHA (a) detection (b) recognition (c) open-set datasets.

A.2 Visualization of Augmentation Attacks

Figure 11 visualizes the augmentation attacks used in adversarial study.

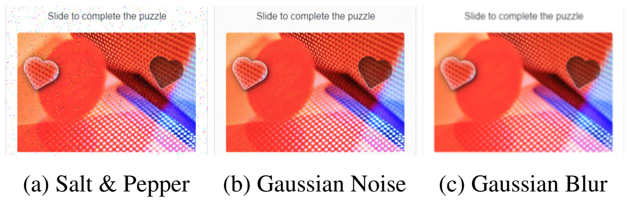


Figure 11: Examples of adversarial augmentations.

A.3 Other Human Verification Methods

This section offers a qualitative analysis of human verification methods apart from CAPTCHA challenges to rate-limit or block visitors.
TLS/SSL Fingerprinting Identify the visitor's web client using TLS and HTTP handshakes, and then present different content for different clients. Countermeasure: handshake impersonation [4].
GeoIP Filtering Restrict access based on visitor's geographical location and IP addresse. Countermeasure: residential proxies [43].
Behavior Analysis Analyze interactions of mouse cursors to distinguish between human users and automated bots. Countermeasure: generate human-like mouse trajectories.
Browser Fingerprinting Exploit JavaScript API to gather data (e.g., screen size, screen orientation, display aspect ratio, device hardware, fonts, plugins, extensions, user-agent) and infer the visitor's identity. Countermeasure: spoofing.