From ML to LLM: Evaluating the Robustness of Phishing Webpage Detection Models against Adversarial Attacks

ADITYA KULKARNI, Indian Institute of Technology (IIT) Dharwad, India
VIVEK BALACHANDRAN, Singapore Institute of Technology, Singapore
DINIL MON DIVAKARAN, Institute for Infocomm Research (I^2R), A*STAR, Singapore
TAMAL DAS, Indian Institute of Technology (IIT) Dharwad, India

Phishing attacks attempt to deceive users into stealing sensitive information, posing a significant cybersecurity threat. Advances in machine learning (ML) and deep learning (DL) have led to the development of numerous phishing webpage detection solutions, but these models remain vulnerable to adversarial attacks. Evaluating their robustness against adversarial phishing webpages is essential. Existing tools contain datasets of pre-designed phishing webpages for a limited number of brands, and lack diversity in phishing features.

To address these challenges, we develop `PhishOracle`, a tool that generates adversarial phishing webpages by embedding diverse phishing features into legitimate webpages. We evaluate the robustness of three existing task-specific models—Stack model, VisualPhishNet, and Phishpedia—against `PhishOracle`-generated adversarial phishing webpages and observe a significant drop in their detection rates. In contrast, a multimodal large language model (MLLM)-based phishing detector demonstrates stronger robustness against these adversarial attacks but still is prone to evasion. Our findings highlight the vulnerability of phishing detection models to adversarial attacks, emphasizing the need for more robust detection approaches. Furthermore, we conduct a user study to evaluate whether `PhishOracle`-generated adversarial phishing webpages can deceive users. The results show that many of these phishing webpages evade not only existing detection models but also users.

Additional Key Words and Phrases: Phishing, Machine Learning, Deep Learning, Large Language Models, Adversarial Attacks

ACM Reference Format:
Aditya Kulkarni, Vivek Balachandran, Dinil Mon Divakaran, and Tamal Das. 2025. From ML to LLM: Evaluating the Robustness of Phishing Webpage Detection Models against Adversarial Attacks. 1, 1 (May 2025), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Phishing attacks extract sensitive information from targeted users. Attackers create phishing webpages and share URLs with users via social media platforms, emails, etc. These requests deceive users into clicking on the URLs and redirect them to phishing webpages where they submit sensitive information. Attackers use this sensitive information to launch identity theft attacks or use information to exploit opportunities for financial profits. In Q1 2024, the APWG [1] recorded 963,994 phishing attacks, with the social media platforms being the most targeted sector, accounting for 37.4% of the attacks. Recent years have witnessed improvements in phishing webpage detection approaches with the use of machine learning (ML) and deep learning (DL) methods.

The ML-based phishing webpage detection solutions [2–5] consist of several stages. Initially, datasets containing both phishing and legitimate samples, including URLs, HTML content, and screenshots, are collected from publicly available repositories. Subsequently, relevant features are extracted based on the URL (such as URL length, presence

XXXX-XXXX/2025/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of @, HTTPS, hyphens in the URL, etc.), webpage content (such as internal and external hyperlinks, CSS styles, pop-up logins, <form action=""> fields, etc.), and third-party features (such as PageRank, Google Index, etc.). Models such as Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM), and many others are trained on these datasets, with performance evaluated using metrics like accuracy, precision, recall, and F1-Score. Finally, the model's efficiency is validated on newly unseen phishing webpages.

With the continual progress in DL techniques, there is an enhanced capability to analyze webpage screenshots and logos by employing state-of-the-art computer vision (CV) models [6–10]. DL techniques are utilized to identify brand logos on webpage screenshots, comparing them with a reference brand list. This comparison considers similarity scores and domain analysis to classify the webpage as phishing or legitimate.

Recently, large language models (LLMs) have gained significant attention and adoption across various industries. These models pretrained on massive datasets are capable to understand and generate text, code, images, and videos. Therefore, the past two years have seen increasing application of LLMs to address different challenges in cybersecurity [11]. In the context of phishing, a recent work by Lee *et al.* [12] proposes an LLM-based pipeline that identifies the targeted brand by analyzing both HTML contents and webpage screenshots. The results show that multimodal large language models (MLLMs) are helpful in outperforming a state-of-the-art solution, namely VisualPhishNet [8], which uses CV for similarity-based phishing detection.

Analyzing existing ML and DL-based phishing webpage detection approaches reveals significant challenges. Initially, researchers face time-consuming efforts for dataset collection containing phishing webpages with diverse phishing features. Furthermore, the short-lived nature of phishing webpages necessitates the extraction of both content-based and third-party-based features. Subsequently, it is essential to evaluate the robustness of these detection models to ensure they can accurately classify new phishing webpages. Recent works [13–17] emphasize the importance of assessing the models against adversarial phishing webpages to determine their performance in detecting such sophisticated threats.

Existing phishing tools (BlackEye [18], ZPhisher [19], and ShellPhish [20]) contain pre-designed phishing webpages datasets that can be used to evaluate the robustness of phishing webpage detection models. However, these datasets are limited to a small number of brands and lack diversity in phishing features. Moreover, these tools cannot generate multiple versions of phishing webpages with diverse sets of phishing features. We tested several brands, such as Yahoo, Amazon, Instagram, Facebook, and Netflix from BlackEye [18]. However, since the tool provides repeated URLs with each execution, many of these URLs were already blocked, rendering the adversarial pages ineffective for assessing evasion attacks.

To overcome these limitations, we develop an automated tool—PhishOracle—that *generates* adversarial phishing webpages by embedding diverse phishing features into legitimate webpages. PhishOracle parses webpage content and randomly selects a set of 12 content-based and 5 visual-based phishing features (refer Table 2), enabling the creation of diverse phishing webpages without being limited to a predefined set of brands. Due to this randomized process, PhishOracle can generate multiple phishing webpages for a single legitimate webpage, each with a unique set of phishing features. Furthermore, PhishOracle is extensible, allowing the integration of new phishing features to adapt to evolving phishing threats. These generated adversarial phishing webpages are used to evaluate the robustness of existing phishing webpage detection models.

In this paper, we evaluate the robustness of phishing detection systems—that use conventional ML (Stack model [21]), DL (VisualPhishNet [8], Phishpedia [9]), and LLM-based phishing detector [12]—against PhishOracle-generated adversarial phishing webpages. Our results reveal that traditional phishing detection models experience a significant drop in detection rate against the adversarial phishing webpages. While LLM-based phishing detector demonstrates higher resilience compared to these models, but it still experiences a decline in detecting adversarial phishing pages. We summarize our contributions:

(1) We propose PhishOracle, a phishing webpage *generator* capable of producing adversarial phishing webpages by randomly embedding content-based and visual-based phishing features into legitimate webpages. We carry out comprehensive evaluations to evaluate the robustness of ML, DL and LLM-based phishing webpage detectors using PhishOracle-generated pages. To the best of our knowledge, this is the first work to do so with one tool.
(2) We conduct a user study to evaluate the effectiveness of PhishOracle-generated phishing webpages generated to deceive users. The experiment results demonstrate that on average ∼48 % of these generated phishing webpages are incorrectly classified as legitimate by the users.
(3) We further validate the effectiveness of adversarial phishing webpages generated by PhishOracle by testing them against 90+ security vendors on VirusTotal.
(4) Finally, we contribute a dataset containing 9,067 legitimate webpage screenshots, on which we manually labelled the identity logos. The PhishOracle code base, web app, generated phishing webpages, and survey screenshots are available on our GitHub repositories [22, 23].

Additionally, we develop PhishOracle web app, which allows users to selectively incorporate both content-based and visual-based phishing features into a legitimate webpage. Compared to the existing phishing webpage dataset tools, PhishOracle offers a broader range of features, including content-based manipulations such as pop-up logins and logo transformation techniques like adjusting opacity, adding watermarks, and more– capabilities that existing tools lack. Table 1 provides a comparative analysis of phishing webpages dataset tools and PhishOracle, highlighting the range of phishing features each tool supports. PhishOracle is a *grey-hat* tool, primarily designed to validate existing phishing webpage detection solutions, which are inherently white-hat in nature. However, its capability to generate phishing webpages introduces a grey-hat aspect to its functionality.

The rest of this paper is structured as follows: Section 2 reviews existing phishing webpage dataset toolkits drawing a comparison with PhishOracle and discusses ML, DL and LLM models for phishing webpage detection. Section 3 defines the threat model, outlining the adversarial capabilities and objectives considered in our evaluation. Section 4 discusses PhishOracle, a tool that generates adversarial phishing webpages by adding randomly selected content-based and visual-based phishing features into legitimate webpages. Section 5 details the performance metrics and datasets used, including the creation of evasion dataset with PhishOracle. It also evaluates robustness of the Stack model, VisualPhishNet, Phishpedia, as well as the LLM-based phishing detector. Moreover, the section presents the evaluation of tools from different security vendors against these adversarial phishing webpages. Section 6 describes a user study assessing the effectiveness of PhishOracle-generated adversarial phishing webpages in deceiving users based on the visual appearance. It also outlines PhishOracle web app, which is designed to generate phishing webpages. Section 7 discusses the limitations of our approach. Finally, Section 8 concludes the paper by discussing the performance of phishing webpage detection models on PhishOracle-generated adversarial phishing webpages.

2 RELATED WORK

In this section, we summarize the existing datasets of pre-designed phishing webpages, which are limited to a small set of brands. We contrast these with PhishOracle, which *generates* phishing webpages by embedding diverse phishing features into legitimate webpages. Additionally, we review existing ML, DL and LLM-based phishing webpage detection solutions.

2.1 Phishing Webpage Datasets

Numerous open-source phishing tools such as BlackEye [18], ZPhisher [19], ShellPhish [20], etc. offer pre-designed phishing webpages tailored to a small number of specific brands. For example, Shellphish [20] has a dataset containing pre-designed phishing webpages for 29 distinct brands. Similarly, ZPhisher [19] has pre-designed

Table 1. Existing Phishing Tools vs PhishOracle

Tools	Phishing Features															Static (S)/ Dynamic (D)
	action="local.php"	Save or Mail Credentials	Added/Removed <script> tags	Local CSS	Local JS	Modified Hypertext References	Font Style	Disable <a> tags	Disable other login <button>	Popup Login	Disabled Right Click	Disabled CTRL and Fn	Body Opacity	Replace blank space with character	Logo Transformations	
BlackEye [18]	✓	✓	✓				✓									**S**
ZPhisher [19]	✓	✓	✓	✓	✓											**S**
ShellPhish [20]	✓	✓	✓	✓	✓	✓	✓									**S**
PhishOracle	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**D**

phishing webpages dataset for 33 brands with each webpage incorporating external resources such as CSS, PHP, and JS files. Consequently, whenever there is a need for a phishing webpage related to any of these brands, the same webpage is consistently fetched from the dataset. These tools exhibit two limitations: a) they cannot *generate* phishing webpages for brands not included in their predefined set, and b) because they already have pre-designed phishing webpages for each brand, they cannot provide a phishing webpage with diverse/new phishing features for the same brand. The tools are inherently *static*, confined to a fixed set of brands.

2.2 Phishing Webpage Detection Approaches

ML and DL techniques have significantly advanced phishing webpage detection by training models on large datasets containing phishing and legitimate samples. These datasets include URLs, webpage content, and screenshots. Phishing samples are obtained from repositories such as PhishTank[1] and OpenPhish[2], and legitimate samples are collected from Common Crawl[3] and Stuff Gate[4]. Alexa[5] and Tranco[24] provide a list of top-ranked legitimate domains, the content of which can be scraped. Datasets like UCI[6], provide feature-extracted files containing both legitimate and phishing samples, including URLs, webpage content, and third-party features. ML models use these features to identify patterns associated with phishing webpages, while DL techniques use neural networks to analyze more complex patterns, such as visual similarity in webpage screenshots and logos.

[1]PhishTank, https://www.phishtank.com/index.php
[2]OpenPhish, https://openphish.com/index.html
[3]Common Crawl, https://commoncrawl.org/
[4]Stuff Gate, https://stuffgate.com.websiteoutlook.com/
[5]Alexa Top Websites, https://www.expireddomains.net/alexa-top-websites/
[6]UCI, https://archive.ics.uci.edu/dataset/327/phishing+websites

2.2.1 ML and DL-based Phishing Webpage Detection Approaches.

Li *et al.* [21] proposed a Stack model (GBDT, XGBoost and LightGBM) for phishing webpage detection using URL and HTML-based features. In this approach, Word2Vec model [25] is used to learn HTML strings and represent them as vector encodings. The Stack model outperforms phishing detection solutions [26–28] by achieving an accuracy of 96.45%. The performance of the Stack model is better than individual ML classifiers for phishing webpage detection.

As DL techniques have evolved alongside the adoption of visual-similarity-based approaches in phishing webpage detection, Fu *et al.* [6] proposed a visual similarity-based approach for phishing webpage detection. The algorithm consists of three stages: a) capturing a screenshot of the suspicious webpage, b) normalizing the screenshot to a fixed size, and c) describing the normalized image to a signature consisting of the pixel color and coordinate features. The Earth Movers' Distance (EMD) algorithm is used to compute the visual similarity between the obtained visual signature and the legitimate webpage signatures and classifies the suspicious webpage as phishing or legitimate. However, this approach performs the classification at the pixel level and does not consider the text. As a result, it cannot detect visually dissimilar webpages.

Afroz *et al.* [7] proposed PhishZoo, a visual-based phishing webpage detection approach based on the webpage profile. A profile of a webpage contains SSL, URL, HTML content and logo features. As a browser loads a webpage, its profile is generated and compared with the stored profiles of legitimate webpages. If the loaded webpage profile does not match with the legitimate SSL, URL, HTML content, and logos, then the user is warned of the webpage possibly being phishing. PhishZoo has advantages over URL-based approaches but also has limitations, such as the approach failing to classify a webpage containing previous logo versions, and if the logos are tilted over 30 degrees.

Similarity-based phishing detection approaches, such as VisualPhishNet [8], identify phishing attempts by analyzing the visual similarity between webpage screenshots. VisualPhishNet utilizes a triplet convolutional neural network (CNN) trained to embed images into a feature space, where visually similar webpages are mapped closer together. The model compares a given webpage screenshot against a set of reference images from known legitimate brands. However, the approach has limitations: it may produce false positives (when different brands have visually similar webpage layouts) and false negatives (when a phishing webpage and its legitimate counterpart exhibit significant design variations despite targeting the same brand). Additionally, the reliance on visual similarity makes the system susceptible to adversarial attacks that modify key visual elements, while maintaining the phishing intent.

Lin *et al.* [9] propose Phishpedia to detect phishing webpages and identify its target brand in a reference brand list. Phishpedia adopts a two-step approach: *object detection* and *brand identification*. The object in this context refers to the *identity logo* present on a webpage screenshot for a given brand. In the first step, the region proposal network (RPN) [29] analyzes the screenshot layout and predicts boxes around logos on the screenshot. The Faster RCNN model, based on the Detectron2 framework (from Facebook [30]), selects the identity logo having the highest confidence score from the set of candidate logos. The next step is logo classification, wherein a ResNetV2 network [31] extracts diverse features from various logo variants belonging to the same brand. The classification task is trained on Logo2K+ dataset [32]. This trained ResNetV2 model is linked with a global average pooling (GAP) layer to generate vectors that represent each logo. These vector representations are compared to logos in the reference brand list using cosine similarity [33]. The brand exhibiting the highest similarity score is identified as the target brand. Upon identifying the brand represented by a logo, the system verifies the legitimacy of the webpage by cross-referencing its domain with the domain associated with the identified brand. Phishpedia outperforms EMD [6], PhishZoo [7] and LogoSENSE [34] in terms of brand identification, phishing detection and runtime overhead.

Building on Phishpedia, Liu *et al.* [10] propose PhishIntention, which enhances phishing detection by incorporating *credential-taking* intention analysis alongside brand identification. Experiment results show that

PhishIntention outperforms EMD [6], VisualPhishNet [8], PhishZoo [7], and Phishpedia [9] in phishing detection, achieving higher precision while at similar recall. Its ability to extract and validate phishing intentions makes it a more effective solution against phishing attacks.

2.2.2 *LLM-based Approaches.*

With technological advancements, LLMs have gained attention in various sectors and their ability to extract contextual information from webpages has made them increasingly valuable in phishing detection. However, existing reference-based phishing detection solutions [9, 10] are constrained by their reliance on reference brand lists, limiting their scalability to a large number of brands. To address this, Li *et al.* [35] propose an LLM-assisted approach that extracts brand information from HTML content and determines whether a webpage attempts to collect user credentials. By not relying solely on logos, this approach expands the phishing detection capability of existing reference-based solutions [9, 10]. In a recent work, Lee *et al.* [12] introduce a two-stage system (requiring no training) that utilizes MLLMs (Gemini, Claude3, and GPT-4) for phishing webpage detection. In the first stage, the MLLM identifies the brand of the webpage by analyzing various aspects of the webpage (such as the HTML content and screenshot). In the second stage, an MLLM verifies whether the identified brand matches with the domain of the webpage. This approach outperforms the existing screenshot-based phishing webpage detection model VisualPhishNet [8].

MLLMs pretrained on massive datasets have the potential to assist or replace the existing models for phishing webpage detection, as they do not require labeled data for training and continuous maintenance (retraining), while still having the capability to understand webpage semantics. Therefore, we take a proactive approach and evaluate an LLM-based phishing detection approach [12] against evasion attacks.

2.2.3 *Robustness of Phishing Detection Models Against Adversarial Attacks.*

Recent works in phishing detection also concentrate on how the phishing detection models perform to classify adversarial phishing evasions. Sabir *et al.* [36] propose URLBUG, an adversarial URL generator designed to generate adversarial phishing URLs by targeting the domain, path and top-level domain parts of URLs. These generated adversarial URLs are used to evaluate the robustness of several ML-based phishing URL detectors. The experimental results reveal that these adversarial URLs significantly challenge the ML-based phishing URL detectors, showing vulnerabilities and limitations.

Object detection models are also prone to adversarial attacks, allowing attackers to create adversarial logos that closely resemble the legitimate ones and evade logo-based phishing webpage detectors [37]. Apruzzese *et al.* [14] examines the real-world impact of adversarial techniques on phishing webpage detection models. Their findings reveal that attackers use simple, low-complexity methods such as blurring, cropping, and masking logos to evade ML-based phishing webpage detectors. Another work by Lee *et al.* [15] generates adversarial logos by adding noise to brand logos in a reference brand list, enabling these logos to evade logo-based phishing webpage detection solutions.

In another work, Apruzzese *et al.* [13] introduce two sets of adversarial attacks: one where an attacker modifies only a few known features and another where the attacker has partial knowledge of the feature set. They develop a robust phishing webpage detection model that withstands these attacks, demonstrating significant performance improvements over existing phishing webpage detection models, which struggle against these adversarial attacks. Ji *et al.* [16] recently performed an extensive assessment of visual similarity-based phishing webpage detection models. The robustness of these phishing detection models is also evaluated against adversarial phishing webpages that utilize visual modifications like rotation, repositioning, resizing, and blurring, showing that even slight modifications can degrade the detection accuracy of the models. To improve the model's robustness, the authors argue to combine text recognition with visual analysis and use preprocessing methods like scaling and denoising to reduce the effects of adversarial alterations.

Inspired by recent efforts to evaluate the robustness of the phishing webpage detection models against adversarial phishing webpages [13–16], we develop a tool that generates such adversarial phishing webpages. This tool aims to enhance phishing webpage detection models by identifying vulnerabilities through adversarial techniques, particularly targeting visual elements such as logos.

3 THREAT MODEL

In this work, we consider a black-box adversary who aims to evade phishing webpage detection models by generating adversarial phishing webpages using widely observed visual transformation techniques, particularly logo manipulations.

The adversary generates adversarial phishing webpages by applying logo-based transformation techniques, which include adjusting opacity, applying Gaussian blur, and overlaying mesh/watermarks. These transformation techniques have been widely used in prior works [14, 16, 38] to evaluate the robustness of visual-based phishing webpage detection models.

The adversary's goal is to generate adversarial phishing webpages targeting brands, such that the generated phishing webpages evade detection. We assume the adversary does not have access to deployed configurations or model weights. This setup aligns with a realistic black-box threat model. The adversary can use PhishOracle to modify the visual appearance of a legitimate webpage and generate adversarial variants embedded with different visual transformations.

Thus, the attack strategy is to download a legitimate webpage for a targeted brand, apply logo transformation techniques, generate adversarial phishing webpages, and evaluate the robustness of the phishing webpage detection models. A successful evasion occurs when the model predicts the brand on the webpage screenshot with low confidence, reducing the likelihood of detection.

Our target systems are phishing webpage detection models that rely on visual-based features to identify the brand in a webpage screenshot and compare it with the domain to detect phishing.

Out of Scope: We assume attackers do not have access to deployed configurations or model parameters. As a result, advanced white-box attacks that require internal model access, such as gradient-based methods (e.g., FGSM [39], DeepFool [40]), are not considered in our work. Similarly, logo perturbation attacks like Generative Adversarial Perturbations (GAP) [15], although feasible in a black-box setting, are beyond the scope of our current work.

4 PROPOSED PHISHING WEBPAGE GENERATION TOOL: PHISHORACLE

Attackers create phishing webpages to collect users' sensitive information. Typically, they modify specific HTML tags like `<form>`, `<a>`, `<link>`, `<script>`, etc., redirecting the acquired sensitive information to databases or email addresses owned by the attackers. In addition to content-based webpage features, maintaining style consistency is another critical aspect that can deceive users. Typically, all webpages in the same organization adhere to a uniform style. The overall style similarity focuses on the visual aesthetics of a webpage, which can be characterized by various CSS design elements such as background colors, text alignments, font family, font size and subtle variations in favicons and logos. When integrated into a legitimate webpage, these visual-based features do not significantly alter its layout, but are sufficient to mislead users into believing that the phishing webpage originates from the legitimate domain owner. The visual resemblance between the generated phishing webpage and the legitimate styling is remarkably close, further deceiving users into perceiving the phishing webpage as a replica of the legitimate one. However, for these phishing webpages to reach users–the actual targets of the attackers–they must first evade phishing webpage detection models. To achieve this, adversarial phishing webpages are created with subtle modifications that allow them to evade phishing webpage detection models, increasing the likelihood that they will reach the users and deceive them.

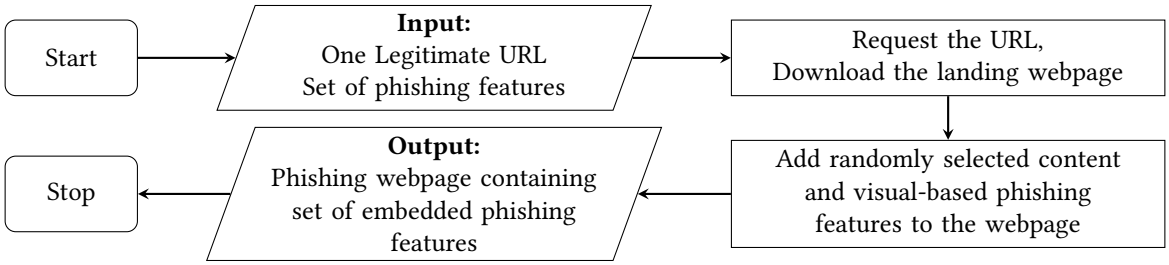
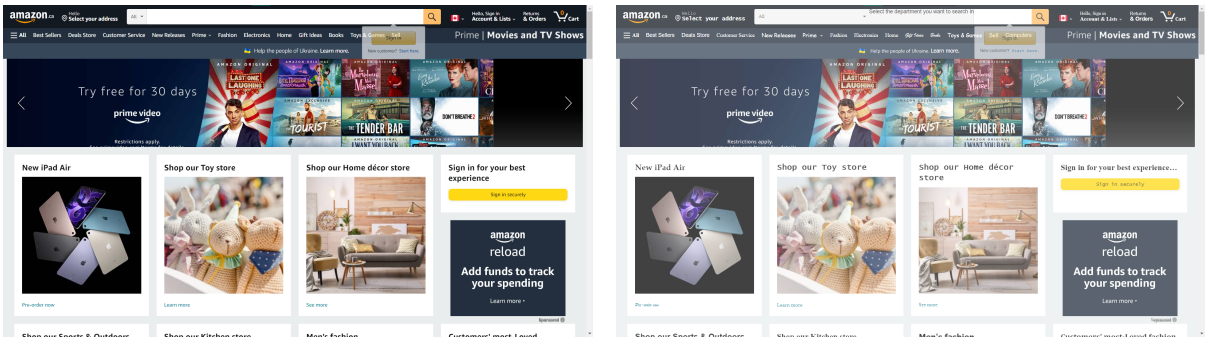


Fig. 1. Phishing Webpage Generation Tool

We propose `PhishOracle` to generate adversarial phishing webpages for any given legitimate webpage by randomly embedding content-based and visual-based features (refer Table 2). Figure 1 depicts the workflow of `PhishOracle`, with two inputs: a legitimate URL and a set/count of phishing features to add to the input legitimate webpage. Firstly, the legitimate URL is requested to fetch and save the webpage content and associated resources. Secondly, the algorithm parses the webpage content to deduce the list of applicable phishing features from Table 2. For example, if the webpage content contains anchor tags (`<a>`) then, relevant phishing features like *C1*, *C3*, *C4*, *C5* and *C10* (refer Table 2) are embedded in the webpage to generate its corresponding phishing webpage.



(a) Legitimate webpage **(b)** Generated Phishing webpage

Fig. 2. `PhishOracle`-generated webpage

Figure 2 shows a legitimate webpage and its corresponding phishing webpage generated by `PhishOracle` by embedding content-based (*C2*, *C5*, *C7*, *C9*, and *C12*) and visual-based (*V1*, *V2*) phishing features as listed in Table 2.

To produce a lookalike URL for the phishing webpage generated by `PhishOracle`, a combination of *homoglyphs*, *prefixes*, and *suffixes* is utilized on the legitimate domain name. This combination results in the creation of domains that closely resemble legitimate domains, followed by corresponding URL parameters. For instance, if we consider the legitimate domain `facebook.com`, this process generates a lookalike domain such as `facebock-login.co`. A selection of homoglyphs examples includes substituting `d` with `cl`, `m` with `nn`, `w` with `vv`, `l` with `1`, `c` with `o`, `m` with `rn`, etc. Additionally, there are prefixes like `secure-`, `logon-`, and `login-`, as well as suffixes such as `-login`, `-logon`, and `-secure`, etc. to enhance the resemblance to legitimate domains.

Table 2. List of Phishing Features based on the Content and Visual Attributes in a Webpage

Type	Naming	Phishing Features	Explanation
Content	*C1*	Hypertext reference	Updating `href` with: `href="#"`, `"#content"`, `"#skip"` or `"Javascript:void(0)"`
	C2	Disable key functions	Restrict users to view source code by disabling *f11* and *Ctrl + U*
	C3	`href` lookalike characters	Alphabets in `href` value are replaced by lookalike characters Ex: replacing `a` with a lookalike character `ā, á, ä`, etc.
	C4	Hide links appearing on status bar	This feature does not allow users to view the `href` link even on hovering on the link
	C5	Disable anchor tags	Does not allow a user to navigate to other pages by clicking on the anchor tags
	C6	Replace blank space with a character	Blank spaces present in the container elements of HTML like `h1, p, span` tags are replaced with characters with `style=color: transparent;`
	C7	Save credentials	The credentials entered in the HTML form input tags are stored in a local file
	C8	Disable other login buttons	Webpages containing login buttons like Google, GitHub, LinkedIn, etc are disabled and the credentials entered in the visible login page are stored locally.
	C9	Pop-up Login	Clicking the login or sign-up button triggers a login page to appear. The action field in the `<form>` tag is then altered, to store the credentials locally
	C10	Pop-up Login by clicking on `a` tags	Clicking the anchor tags, opens up a login page and the credentials are stored locally
	C11	IFrame tag with login page	`<iframe>` tags are added with a login page, the credentials entered are stored locally
	C12	Add dummy tags	Dummy `<img>`, `<link>`, `<script>`, `<a>` and `<div>` tags are added to increase the DOM structure of webpage
Visual	*V1*	Body Opacity	CSS `opacity` adjusts webpage transparency from 0 to 1.
	V2	Text Styling	The `font-family` changes the style of webpage text
	V3	Opacity on Logo	CSS `opacity` property makes the logo image transparent. `<img>` tags containing images with `.png` and `.svg` extensions are considered and a `opacity` of 0.1 to 0.35 is added to make the logo transparent
	V4	Adding Watermark on logo	A watermark is added either on the bottom right corner or diagonally on logo image
	V5	Image Transformations	Techniques like adding (1) clockwise or anti-clockwise rotation, (2) Gaussian blur, (3) grey-mesh and (4) noise, transforms the logo image

5 PERFORMANCE EVALUATION

In this section, we outline the performance metrics used to evaluate phishing webpage detection models. We detail the clean dataset used for model training and the evasion dataset – containing adversarial phishing webpages generated by PhishOracle – to evaluate the robustness of these models. Additionally, we discuss the evaluation of security tools against these adversarial phishing webpages.

5.1 Performance Metrics

The fundamental metrics to evaluate classifiers are True Positive (*TP*), False Positive (*FP*), True Negative (*TN*), and False Negative (*FN*). In the context of the phishing detection models, these metrics are used for determining the count of actual to predicted values as shown in Table 3. The metrics *TP, FP, TN* and *FN* are elaborated as follows:

- True Positive (*TP*): The model correctly identifies a phishing website as phishing.
- False Positive (*FP*): The model incorrectly identifies a legitimate website as phishing.
- True Negative (*TN*): The model correctly identifies a legitimate website as legitimate.
- False Negative (*FN*): The model incorrectly identifies a phishing website as legitimate.

Table 3. Confusion Matrix

		Predicted	
		Phishing	*Legitimate*
Actual	*Phishing*	True Positive	False Negative
	Legitimate	False Positive	True Negative

These metrics are used to compute the *Accuracy* ($\frac{TP+TN}{TP+FP+TN+FN}$), *Precision* ($\frac{TP}{TP+FP}$), *Recall* ($\frac{TP}{TP+FN}$) and *F1-Score* ($\frac{2\times Precsion\times Recall}{Precision+Recall}$) of classifiers.

5.2 Dataset Description

With technological advancements, attackers find new features to evade existing phishing webpage detection solutions. Various solutions [21, 26–28] consider URL and HTML features, and a few solutions [6–9] use visual similarity features to classify new phishing webpages. With the current research focus on evaluating the robustness of phishing webpage detection models against adversarial phishing webpages, we evaluate existing models for phishing webpage detection using adversarial phishing webpages generated by PhishOracle.

In our study, we select the Stack model [21] due to its effectiveness in ML-based phishing webpage detection, to assess its capability to classify PhishOracle-generated adversarial phishing webpages that include a variety of phishing features. Next, we select VisualPhishNet [8], a screenshot-based phishing webpage detection model that analyzes the visual similarity between webpage screenshots to distinguish phishing webpages and legitimate ones by learning discriminative features based on brand identity and layout structure. Moreover, we select Phishpedia [9], known for its robustness in phishing webpage detection and identifying target brands in webpage screenshots. This makes it ideal for identifying brands in adversarial phishing webpage screenshots generated by PhishOracle, incorporating different logo transformation techniques. Finally, we select an LLM-based phishing detector [12], selecting Gemini 1.5 Flash [41] as the LLM, to evaluate the solution's capability in identifying the target brand in HTML content and webpage screenshots of PhishOracle-generated adversarial phishing webpages, and verify whether the identified brand matches with the domain of the webpage.

Table 4. Dataset Collection Summary

Notation	Description	Collection Period	Sample Size	
			Phishing	Legitimate
$\mathcal{T}^1$	Contains URL, HTML content, and webpage screenshots of phishing and legitimate samples	Dec'23 to Jan'24	9,105	8,727
$\mathcal{T}^2$	Contains URL, HTML content, and webpage screenshot of legitimate samples, with manually annotated logos required to train the Faster RCNN model in the Phishpedia experiment	May'24	0	9,067

As the datasets to train these selected models [8, 9, 21] are outdated, we retrain them using our latest datasets (see Table 4) and evaluate their robustness in detecting the adversarial phishing webpages generated by `PhishOracle`. We collect these dataset samples by using a Python script that includes various libraries, such as `Selenium`, `requests`, and `urllib.parse`. This script scraped 9,105 phishing samples from PhishTank[1], and 8,727 legitimate samples from Tranco [24], collected between December 2023 and January 2024. This collected dataset is referred to as $\mathcal{T}^1$ in Table 4. However, some brands in the reference brand list of the Phishpedia experiment prohibited downloading the webpage content but allowed capturing screenshots, while others prohibited both. To ensure their inclusion in the dataset, we manually captured screenshots of these websites. As a result, we collected a total of 9,067 legitimate webpage screenshots, referred to as $\mathcal{T}^2$ in Table 4. Furthermore, we manually annotated the brand logos in each of the 9,067 webpage screenshots to train the Faster RCNN model for brand identification in the Phishpedia experiment.

The collected samples in Table 4 are categorized into *CleanSet*, and *EvasionSet*, each serving a distinct purpose for training and evaluating the robustness of the selected phishing webpage detection models. Table 5 describes the categories of these *CleanSet*, and *EvasionSet* datasets. *CleanSet*^{1} corresponds to the $\mathcal{T}^1$ dataset (refer Table 4), which is used to train the Stack model [21] and to evaluate LLM-based phishing detector [12]. It includes 9,105 phishing samples and 8,727 legitimate samples, both containing URL, HTML content, and webpage screenshots. *CleanSet*^{2} corresponds to the $\mathcal{T}^2$ dataset (refer Table 4), primarily used to train the Faster RCNN model for brand identification in the Phishpedia experiment [9]. It consists of 9,067 legitimate samples. Moreover, 909 phishing samples ($\subset \mathcal{T}^1$ dataset) and 82 legitimate samples ($\subset \mathcal{T}^2$ dataset), are specifically used to evaluate the performance of the brand identification and phishing detection of the entire Phishpedia system [9], as well as the LLM-based phishing detector [12]. *CleanSet*^{3} is used to train the VisualPhishNet model [8]. It consists of 896 phishing samples ($\subset \mathcal{T}^1$ dataset) covering 41 target brands, ~600 legitimate webpage screenshots, and ~300 additional legitimate webpage screenshots (from non-target brands $\subset \mathcal{T}^2$ dataset). This dataset is used to train the triplet CNN model for visual similarity-based phishing detection.

The *EvasionSet* datasets in Table 5—containing adversarial phishing webpages generated by `PhishOracle`—evaluate the trained models. *EvasionSet*^{1} contains 1,000 legitimate samples ($\subset \mathcal{T}^1$ dataset) with corresponding 1,000 adversarial phishing webpages generated using `PhishOracle`, and is used to evaluate the Stack model [21], and the LLM-based phishing detector [12] (referred as *LLM-PD*^{H} in Table 6). *EvasionSet*^{2} consists of 82 legitimate samples ($\subset \mathcal{T}^2$ dataset), selected from Phishpedia's reference brand list, along with 170 adversarial phishing webpages generated using `PhishOracle`. This dataset is used to evaluate the Phishpedia system [9], as well as the LLM-based phishing detector [12] (referred as *LLM-PD*^{S} in Table 6). *EvasionSet*^{3} contains 107 phishing and 41 legitimate samples, which are a subset of *EvasionSet*^{2} and focus on the 41 target brands in VisualPhishNet [8].

Table 5. *CleanSet*, and *EvasionSet* Description for Training and Evaluating the Phishing Webpage Detection Models

Notation	Description	Sample Size		
		Phishing	Legitimate	PhishOracle (generated phishing)
CleanSet^1	$\mathcal{T}^1$ dataset used to train the Stack model [21] and to evaluate the LLM-based phishing detector [12]	9,105	8,727	0
CleanSet^2	$\mathcal{T}^2$ dataset used to train the Faster RCNN model, and ∼900 phishing ($\subset \mathcal{T}^1$), and 82 legitimate samples ($\subset \mathcal{T}^2$), are used to evaluate the performance of brand identification and phishing detection model of the entire Phishpedia system [9], and the LLM-based phishing detector [12]	909	9,067	0
CleanSet^3	∼900 phishing samples ($\subset \mathcal{T}^1$), and ∼600 legitimate samples covering the 41 target brands + ∼300 legitimate webpage screenshots ($\subset \mathcal{T}^2$), from non-target brands, used to train the triplet CNN model in the VisualPhishNet model [8]	896	911	0
EvasionSet^1	Contains adversarial phishing webpages generated by PhishOracle for legitimate webpages selected from $\mathcal{T}^1$ dataset to evaluate the Stack model [21], and LLM-based phishing detector [12]	0	1,000	1,000
EvasionSet^2	Contains 170 adversarial phishing webpages generated by PhishOracle for 82 legitimate webpages ($\subset T^2$ dataset) in the reference brand list of Phishpedia [9] and to evaluate the performance of the Phishpedia [9] and the LLM-based phishing detector [12]	0	82	170
EvasionSet^3	Contains 41 legitimate and 107 corresponding adversarial phishing webpages ($\subset$ *EvasionSet*^{2}), and used to evaluate the performance of VisualPhishNet [8]	0	41	107

This dataset is used to assess the robustness of the VisualPhishNet model against adversarial phishing webpages generated by PhishOracle.

5.3 Generating PhishOracle Validation Dataset

Developed using Python version 3.10.2, our PhishOracle tool uses various libraries, including BeautifulSoup, requests, webbrowser, PIL, and urllib.parse. For the Stack model experiment, we select 1,000 legitimate webpages ($\subset \mathcal{T}^1$ dataset) and embed relevant content-based and visual-based phishing features (refer Table 2) to generate corresponding 1,000 phishing webpages, forming the *EvasionSet*[1] dataset. The algorithm includes a mechanism for detecting relevant tags within the webpages. The detection of such tags triggers the embedding of corresponding phishing features into the webpage content. For instance, if the algorithm identifies <a> tag within the legitimate webpage, it embeds features such as *C1*, *C3*, *C5*, *C10* and *C12* (refer Table 2), resulting in the generation of the corresponding phishing webpage. This entire process ensures the creation of balanced sets comprising legitimate and corresponding phishing webpages.

For VisualPhishNet, Phishpedia, and LLM-based phishing detection experiments, we select 82 legitimate webpages ($\subset \mathcal{T}^2$ dataset), all belonging to Phishpedia's reference brand list. We then generate 170 corresponding adversarial phishing webpages by incorporating visual-based features on logos (refer Table 2). Note that a single webpage can have multiple PhishOracle-generated phishing webpages, each generated by embedding different visual-based features, specifically by incorporating logo transformation techniques (refer to visual-based features in Table 2). The *EvasionSet*[2] dataset comprises screenshots of these 82 legitimate and 170 generated phishing webpages, which are used to evaluate Phishpedia and LLM-based phishing detector. A subset of *EvasionSet*[2], referred to as *EvasionSet*[3], is constructed by selecting phishing and legitimate samples associated with the 41 target brands of VisualPhishNet. This dataset is specifically designed to assess the robustness of the VisualPhishNet model against phishing attempts targeting protected brands. Furthermore, we generate lookalike URLs for each of these generated phishing webpages (in *EvasionSet*[1], *EvasionSet*[2], and *EvasionSet*[3]) using homoglyphs, suffixes, and prefixes. The source code and the datasets used in this experiment are available on our GitHub repository [22].

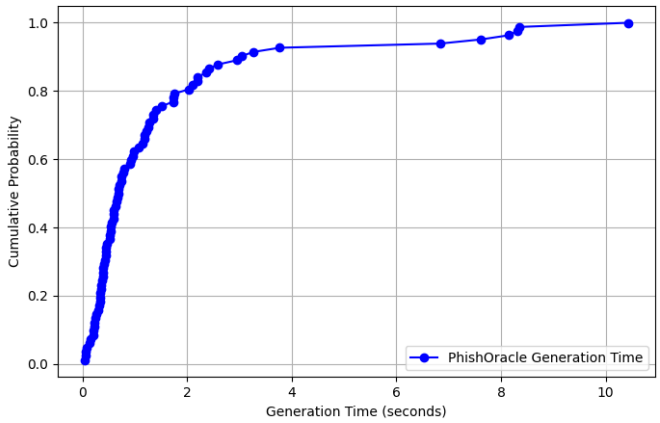


Fig. 3. CDF of PhishOracle processing time for generating adversarial phishing webpages in the *EvasionSet*[2] dataset

Figure 3 shows the performance evaluation of PhishOracle on *EvasionSet*[2] dataset. The average processing time taken to embed a diverse set of content and visual-based phishing features into legitimate webpages is ~1.48 seconds. The majority of the webpages are processed under 3 seconds, with only a few outliers extending up to 10 seconds due to potentially complex webpage structures. The CDF plot indicates that around 80% of webpages were processed in under 3 seconds, demonstrating the efficiency of PhishOracle to generate adversarial phishing webpages.

5.4 Validating Phishing Webpage Detection Models

In this section, we discuss the experiment setup for Stack model [21], VisualPhishNet [8], Phishpedia [9] and LLM-based phishing detector [12], and illustrate their performance on detecting adversarial phishing webpages generated by `PhishOracle`.

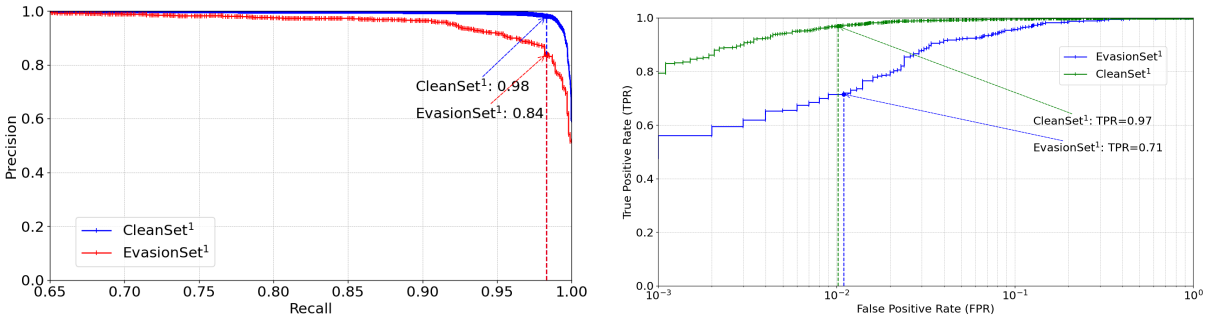
5.4.1 Stack Model.

Li *et al.* [21] propose a Stack model for phishing webpage detection using URL and HTML features, without considering third-party-based features (like DNS records, web traffic, etc.).

Setup: To carry out our experiments, we use the feature extraction code of Stack Model made available by Phishpedia's authors on their GitHub repository [42] and we develop the Stack model using GBDT, XGBoost and LightGBM classifiers.

Training & Testing the model: We use *CleanSet*[1] containing URLs and HTML contents of 9,105 phishing and 8,727 legitimate webpages; and the train:test split is set to 80:20. In our experiments, Stack model achieves a high detection rate (recall) of ~98.32% at a high precision of ~98.67% on the*CleanSet*[1] dataset.

Validation on *EvasionSet*[1]: For evaluating the robustness of the Stack model, we now use the *EvasionSet*[1] that contains URLs and HTML content of legitimate and `PhishOracle`-generated phishing webpages. Stack model's detection rate falls to ~70.86% for a comparable precision of ~98.61%. Observe that, this is a significant reduction in comparison to the performance achieved on the test set *CleanSet*[1] (as mentioned above)—~28% in recall for about the same precision. Therefore, the Stack model is not robust to simple feature modifications (refer Section 4), and confirms the observations made in a recent study [43].



(a) Precision vs Recall

(b) TPR vs FPR

Fig. 4. Precision vs Recall (Figure 4a) and TPR vs FPR (Figure 4b) for the Stack Model

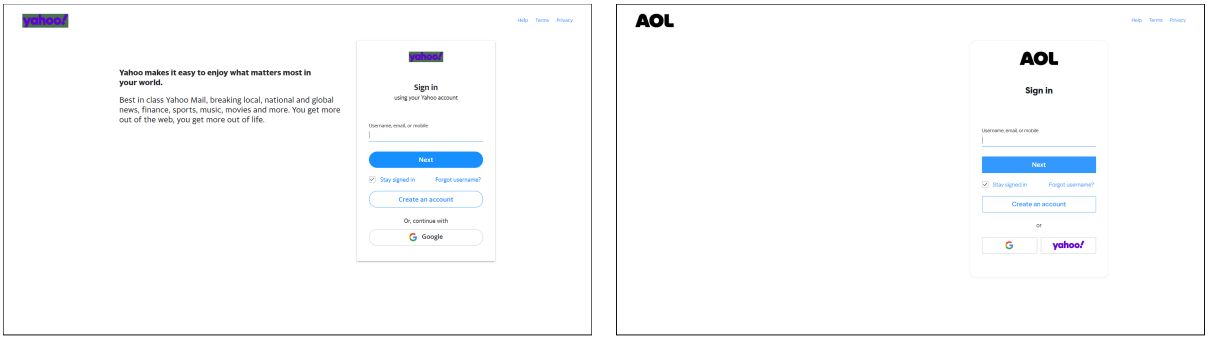
Figure 4a displays the precision-recall curve, illustrating the detection efficacy of the Stack model on the *CleanSet*[1] and *EvasionSet*[1] datasets. For a recall of ~98%, the Stack model's precision drops significantly on the *EvasionSet*[1] to ~84% in contrast to ~98% on the *CleanSet*[1]. This performance reduction indicates the model's reduced effectiveness in detecting adversarial phishing webpages generated by `PhishOracle`. In addition, Figure 4b displays the ROC curve for both datasets, drawn on a logarithmic scale to show the model's performance at low false positive rates (FPR). For a fixed FPR of $\sim 10^{-2}$, the model achieves a true positive rate (TPR) of ~97% on the *CleanSet*[1], which drops to ~71% on the *EvasionSet*[1], indicating reductions in both precision and TPR for adversarial phishing webpages.

5.4.2 VisualPhishNet.

Abdelnabi *et al.* [8] propose VisualPhishNet to detect phishing webpages by analyzing the visual similarity between webpage screenshots using a triplet CNN model.

Setup: We initiate the experiment by cloning the baseline model from the Phishpedia GitHub repository [42].

Training and Testing Model: In this experiment, we use phishing and legitimate webpage screenshot datasets. The *CleanSet*^{3} dataset consists of ~900 phishing samples ($\subset \mathcal{T}^1$ dataset) targeting 41 brands, along with ~600 legitimate webpage screenshots for these 41 target brands, collected by visiting their official website hyperlinks. Additionally, to ensure an unbiased training process, we include ~300 legitimate webpage screenshots from non-target brands ($\subset \mathcal{T}^2$ dataset). This dataset is used to train the triplet CNN model in VisualPhishNet for visual similarity-based phishing detection. We follow the 60:40 train-test split, as suggested in [8], to evaluate the model's performance effectively. The retrained model achieves a precision of ~98.42% at a recall of ~88.35% on the *CleanSet*^{3}.



(a) Adversarial Phishing Webpage targeting Yahoo **(b)** Legitimate Webpage of AOL

Fig. 5. VisualPhishNet misclassify adversarial phishing webpage targeting Yahoo (Figure 5a) as AOL (Figure 5b) due to high similarity in their visual appearance

Validation on *EvasionSet*^{3}: Finally, we evaluate the trained VisualPhishNet model on the *EvasionSet*^{3}, containing 41 legitimate and 107 adversarial phishing webpages generated by `PhishOracle`. This results in a considerably lower phishing detection rate (recall) of ~72.89% at a precision of ~98.73%. This demonstrates that the adversarial phishing webpages generated by `PhishOracle`, which incorporate logo transformation techniques (such as watermark, rotation, opacity, etc.), evade the VisualPhishNet model. For example, as shown in Figure 5, the model incorrectly identifies an adversarial phishing webpage targeting Yahoo (Figure 5a) as AOL (Figure 5b), due to similarities in the visual appearance of the webpage layouts, including logo placement and form fields.

Observations: To evaluate the robustness of VisualPhishNet, we use *EvasionSet*^{3}, containing adversarial phishing webpages generated by `PhishOracle`. The detection rate of VisualPhishNet drops to ~72.89% for a comparable high precision of ~98.73% (refer Table 6). Observe that this is a significant reduction compared to its performance on the test set *CleanSet*^{3} (as mentioned above)–~16% decrease in recall for about the same precision. This indicates that VisualPhishNet is affected by the logo-based adversarial manipulations. These findings align with recent evaluations [16], showing that the screenshot-based models like VisualPhishNet struggle with accurate brand identification when logos are altered, removed or replaced.

5.4.3 Phishpedia.

Lin *et al.* [9] propose Phishpedia to detect phishing webpages targeting a specific brand in a reference brand list.

Setup: We initiate the experiment by cloning the Phishpedia GitHub repository [42]. To carry out this experiment on Phishpedia, we manually labelled identity logos on 9,067 legitimate webpage screenshots in the $\mathcal{T}^2$ dataset.

Training & Testing the model: In this experiment, we consider two datasets. For evaluating the object detection task, we use $\mathcal{T}^2$ dataset (described in Table 4), containing 9,067 legitimate URLs and webpage screenshots, and split it into 7,539 samples for training and 1,528 for testing. This results in a train:test split of 83:17. For the brand identification task, which uses the Siamese model pretrained on Logo2K+ dataset [32], and the phishing detection task, we use *CleanSet*^{2}, containing 909 phishing webpage screenshots with manually labelled identity logos and 82 legitimate webpage screenshots from the reference brand list, with a train:test split of 80:20.

Results: The logo detection task achieves the mean Average Precision (mAP) [44] of ~43.42%, which is considered good in object detection. The brand identification model accurately detects the brand on ~97.77% of phishing webpages. In phishing detection, the entire Phishpedia system achieves a precision of ~97.25% at a recall of ~81.63%. The values are slightly lower than the precision of ~98.2% at a recall of ~87.1% reported in the paper [9]; the reasons could be that, we have a smaller dataset for training in our work, or there is a difference in the quality of data. However, close to ~81.63% recall at a high precision of ~98% is still a good performance.

Validation on *EvasionSet*^{2}: Finally, we evaluate the newly trained Phishpedia system on the *EvasionSet*^{2}, containing 82 legitimate webpage screenshots and 170 `PhishOracle`-generated adversarial phishing webpage screenshots. This results in a precision of ~76.40% at a recall of ~40% for phishing detection (refer Table 6). Observe that the `PhishOracle`-generated adversarial phishing webpage screenshots containing logo transformation techniques (such as adding watermark, noise, etc.), can evade brand identification in the Phishpedia model.

5.4.4 Multimodal LLMs.

With the emergence of LLMs, we today have several MLLMs that can process, analyze and generate texts, images and other multimedia contents. The capabilities of the MLLMs to analyze texts and images make them suitable for our task of brand identification in HTML content and webpage screenshots. In a recent work, Lee *et al.* [12] propose a two-stage system of MLLMs for phishing webpage detection. In the first stage, an MLLM analyzes various aspects of webpage, including its HTML content and screenshots, to identify the brand. In the second stage, an MLLM verifies whether the identified brand matches the domain of the webpage (URL), flagging mismatches as phishing. The approach employs separate prompts for analyzing HTML content and webpage screenshots to identify the brand, along with an additional prompt to verify whether the identified brand matches the domain of the webpage. In our study, we adopt these prompts to evaluate the LLM-based phishing detection pipeline [12] using Gemini 1.5 Flash [41] on both clean and adversarial phishing webpages, using *CleanSet*^{1} and *CleanSet*^{2}; and *EvasionSet*^{1} and *EvasionSet*^{2}, respectively.

To ensure a fair comparison with existing phishing detection models, we utilize two Gemini-based variants: *LLM-PD*^{H} and *LLM-PD*^{S} from [12]. The Stack model, which relies on URL and HTML-based features, is evaluated on *EvasionSet*^{1}. Accordingly, we use the HTML prompt from [12] and provide the corresponding HTML contents as input to the LLM-based pipeline, referring to this variant as *LLM-PD*^{H}. Similarly, Phishpedia, which identifies brands in webpage screenshots is evaluated on *EvasionSet*^{3} – so we use the screenshot prompt from [12] and input the corresponding webpage screenshots into the LLM-based pipleine, referring to this as *LLM-PD*^{S}. This approach ensures that the performance of LLM-based approach is directly compared to state-of-the-art phishing webpage detection models on the same evasion datasets.

(1) Brand identification from **HTML content**

Setup: This experiment evaluates the ability of the LLM-based pipeline from [12] using Gemini [41] to identify brands from the HTML content of webpages. We employ the HTML-based prompt from [12] and refer to this variant as *LLM-PD*^{H} (as shown in Table 6). The model is tested on the *CleanSet*^{1} dataset, which comprises phishing and legitimate samples. We evaluate the performance of *LLM-PD*^{H} in brand identification and verifying its consistency with the domain of the webpage. The model achieves a precision of ~76.41% at a recall of ~73.83%, and an F1-Score of ~75.09%.

Robustness Evaluation on *EvasionSet*^{1}: To assess the robustness of *LLM-PD*^{H} against adversarial phishing webpages, we evaluate its performance on *EvasionSet*^{1}, which contains phishing samples generated by `PhishOracle`.

Results: On *EvasionSet*^{1} dataset, *LLM-PD*^{H} achieves a precision of ~85.16% at a recall of ~67.55%, and an F1-Score of ~75.34%.

Observations: *LLM-PD*^{H} achieves a precision of ~85.16% at a recall of ~67.55%, and an F1-Score of ~75.34% on *EvasionSet*^{1}, indicating a decline in recall compared to the *CleanSet*^{1}. This indicates an increase in false negatives, where more phishing webpages are misclassified as legitimate. However, this corresponding increase in precision offsets this drop, resulting in a comparable F1-Score. These results suggest that the LLM-based phishing detection pipeline remains relatively robust against adversarial phishing attacks, though further improvements may be necessary to enhance recall.

(2) Brand identification from **webpage screenshots**

Setup: In this experiment, we evaluate the ability the LLM-based phishing pipeline [12] to identify brands from webpage screenshots. We use the screenshot-based prompt from [12] and refer to this variant of Gemini as *LLM-PD*^{S} (as shown in Table 6). The model is tested on the *CleanSet*^{2}, a dataset consisting of both legitimate and phishing webpage screenshots. We analyze how effectively *LLM-PD*^{S} identifies brand names and verifies them against the domain of the webpage. The model achieves an impressive precision of ~98.83% at a recall of ~97.25%, and an F1-Score of ~98.03%.

Robustness Evaluation on *EvasionSet*^{2}: To assess the robustness of *LLM-PD*^{S}, we evaluate its performance on *EvasionSet*^{2}, which contains screenshots of legitimate webpages along with their corresponding adversarial phishing webpages variants generated by `PhishOracle`.

Results: Evaluation of *LLM-PD*^{S} on *EvasionSet*^{2} results in a precision of 100% at a recall of ~79.51%, with an F1-Score of ~88.59%.

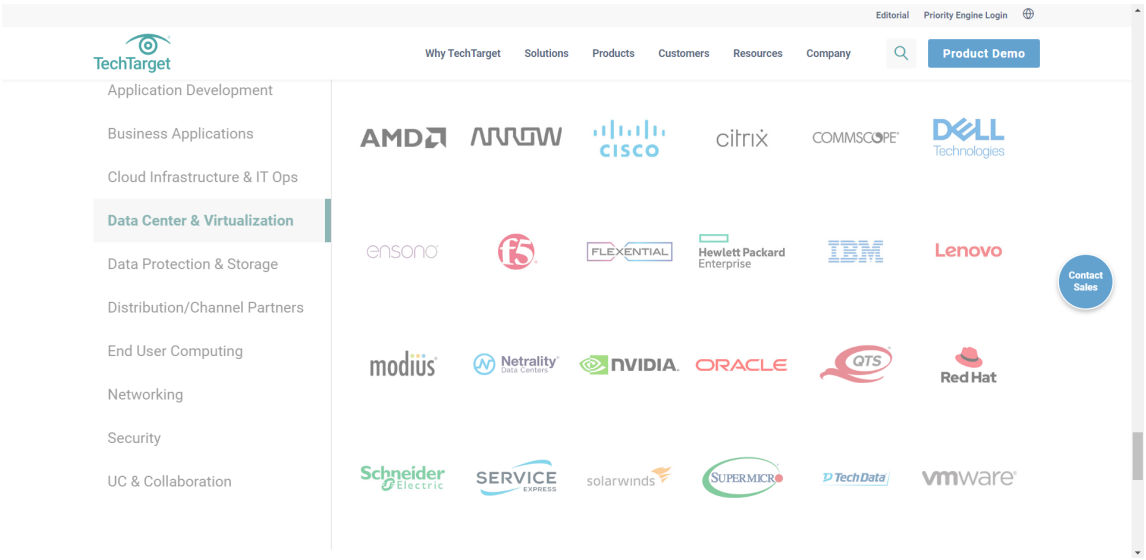


Fig. 6. Webpage Screenshot containing multiple logos of different brands

Observations: *LLM-PD*^{S} is robust on the adversarial phishing webpages generated by `PhishOracle` but it experiences a drop of ~10% F1-Score on *EvasionSet*^{2} as compared to *CleanSet*^{2}. For instance, in an

adversarial phishing webpage targeting Yahoo (see Figure 5a), the first-stage MLLM correctly identifies the brand and provides supporting evidence. The second-stage MLLM while comparing the identified brand with the domain name / URL (`http://signup-yahoo.com`) incorrectly associates it with the identified brand as belonging to the same company, leading to evasion. Nevertheless, *LLM-PD*^{S} outperforms VisualPhishNet [8] and Phishpedia [9] in effectively identifying brands on adversarial phishing webpages generated by `PhishOracle`. Furthermore, *LLM-PD*^{S} efficiently identify the actual brand name in webpage screenshots containing logos of multiple brands. As an illustration, it correctly identifies the brand name as 'TechTarget' in the webpage screenshot containing logos of multiple brands, as shown in Figure 6. Once the brand is correctly identified, comparing the URL's domain name with the identified brand's domain name gives the detection result.

Table 6. Performance of Stack Model, VisualPhishNet, Phishpedia, and *LLM-PD*^{H}, and *LLM-PD*^{S}

Model	*CleanSet*				*EvasionSet*			
	Dataset	*Precision*	*Recall*	*F1-Score*	**Dataset**	*Precision*	*Recall*	*F1-Score*
Stack Model [21]	*CleanSet*^{1}	98.67%	98.32%	98.49%	*EvasionSet*^{1}	98.61%	70.86%	82.46%
VisualPhishNet [8]	*CleanSet*^{3}	98.42%	88.35%	93.11%	*EvasionSet*^{3}	98.73%	72.89%	83.87%
Phishpedia [9]	*CleanSet*^{2}	97.25%	81.63%	88.76%	*EvasionSet*^{2}	76.40%	40%	52.51%
LLM-PD^H	*CleanSet*^{1}	76.41%	73.83%	75.09%	*EvasionSet*^{1}	85.16%	67.55%	75.34%
LLM-PD^S	*CleanSet*^{2}	98.83%	97.25%	98.03%	*EvasionSet*^{2}	100%	79.51%	88.59%

Table 6 provides a comprehensive comparison of the Stack model, VisualPhishNet, Phishpedia, and *LLM-PD*^{H}, and *LLM-PD*^{S} in phishing webpage detection.

Takeaways.

i) The detection rate of the Stack model [21] drops by around 28% on `PhishOracle`-generated *EvasionSet*^{1} compared to its performance on *CleanSet*^{1}, when evaluated at a comparable precision.

ii) VisualPhishNet [8] experiences a drop of around 16% in detection rate when evaluated on *EvasionSet*^{3} relative to its performance on *CleanSet*^{3}, when tested on a comparable precision.

iii) Phishpedia [9] suffers a drastic drop of approximately 40% in detection rate on *EvasionSet*^{2} due to adversarial logo transformations (refer to Table 2), at a 20% lower precision. This indicates that transformed logos can evade Phishpedia. Another contributing factor could be the smaller training dataset used in our work.

iv) Compared to other phishing detectors, LLM-based phishing detector (*LLM-PD*^{S}) (with no task-specific training) is more robust against adversarial phishing webpages generated by `PhishOracle`. Nonetheless, it also experiences a decline of performance—around 10% in F1-Score on *EvasionSet*^{2} compared to *CleanSet*^{2}.

5.5 Evaluation of Security Tools against Adversarial Phishing Attacks

In this section, we present a 9-day evaluation of adversarial phishing webpages generated by `PhishOracle`, focusing on their detection by security tools.

Setup: We conducted a 9-day study to evaluate effectiveness of the security tools in detecting the adversarial phishing webpages generated by `PhishOracle`. The evaluation focused on VirusTotal [45], with more than 90 security vendors, along with GSB and Microsoft Defender SmartScreen. We initiate the experiment by generating one adversarial phishing webpage for eight brands, including Microsoft, Yahoo, Store Steam, TalkTalk Group, UIBK, UChile, Trane, and LibreTexts. These webpages are designed by embedding diverse content-based features – such as altering form action fields, embedding pop-up login forms, using Javascript-based navigation prevention

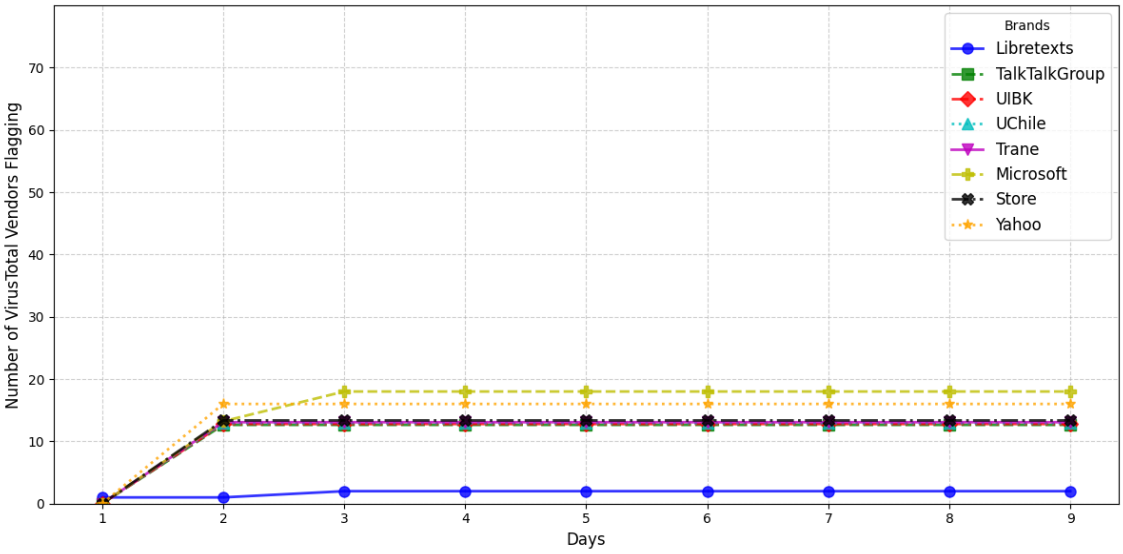


Fig. 7. Evaluation of Security Tools Over 9 Days Against Adversarial Phishing Attacks

to restrict user movement – and visual-based features, including opacity, and logo transformation techniques (refer Table 2). Once generated, these adversarial phishing webpages are hosted on GitHub Pages, allowing them to be publicly accessible for evaluation.

Evaluation Methodology: The adversarial phishing webpages generated by `PhishOracle` were examined over a 9-day span at 24-hour intervals. The total number of security vendors in VirusTotal flagging these webpages as phishing or suspicious were documented. Additionally, the identification by GSB and Microsoft Defender SmartScreen were also tracked to observe how promptly these popular security solutions recognize these adversarial phishing webpages.

Results: With a total of 96 security vendors on VirusTotal, the detection rates varied significantly among different brands. On Day 1, only one phishing webpage targeting LibreTexts, was flagged by only one security vendor (Trustwave). The other phishing websites were still not marked by any of the security vendors. On Day 2, the Microsoft phishing webpage was flagged by 6 security vendors on VirusTotal, and the Microsoft Defender SmartScreen marked it as "Dangerous Site". Several other phishing webpages targeting brands such as TalkTalk Group, UIBK, UChile, Trane, Steam, and Yahoo were flagged by 13 vendors, such as GSB, Trustwave, Webroot, and CyRadar. The phishing webpage from LibreTexts was marked as "Dangerous" by GSB. On Day 3, the number of detections rose for phishing webpages targeting popular brands such as Microsoft and Yahoo, which were flagged by 18 and 16 security vendors respectively. Moreover, the LibreTexts phishing webpage was marked as "Dangerous" by Microsoft Defender SmartScreen, however, it remained unflagged by most of the security vendors on VirusTotal. From Days 4 to 9, the count of vendors marking these hosted adversarial phishing webpages remained stable. Figure 7 displays the detection rates of security vendors against adversarial phishing webpages generated by `PhishOracle`.

Observations: The 9-day evaluation study showed that phishing webpages targeting well-known brands (e.g., Microsoft, Yahoo) were flagged by more security vendors than those targeting lesser-known brands (e.g., LibreTexts). However, detections were alarmingly slow, with the majority of security vendors failing to identify

phishing webpages within the first 24 hours. For instance, the Microsoft phishing webpage was flagged by only six vendors by Day 2, meaning over 90% of the 96 VirusTotal security vendors failed to detect it within the first day. Similarly, the LibreTexts phishing webpage remained largely undetected, with just one vendor identifying it by Day 1.

These results align with prior research [46], which reported an average phishing site lifespan of 2.25 days, showing that phishing webpages remain active long enough to deceive victims. Detection rates in our study followed a similar trend, where most phishing webpages were flagged only after two or more days, and detections stagnated after Day 3. This delay is particularly concerning because even a few hours of undetected activity provides ample time for attackers to successfully execute phishing campaigns.

Our findings further support previous work [47], which highlights inconsistencies in how VirusTotal vendors classify phishing URLs. While some vendors flagged phishing webpages within two days, most security solutions failed to detect them within the first 24 hours, exposing critical detection gaps. Even for high-profile brands like Microsoft and Yahoo, only 16 to 18 out of 96 security vendors flagged them by Day 3, meaning ~80% of security vendors still failed to detect them by this time. This detection gap left phishing webpages accessible to potential victims for an extended period.

6 USER STUDY AND PHISHORACLE WEB APP

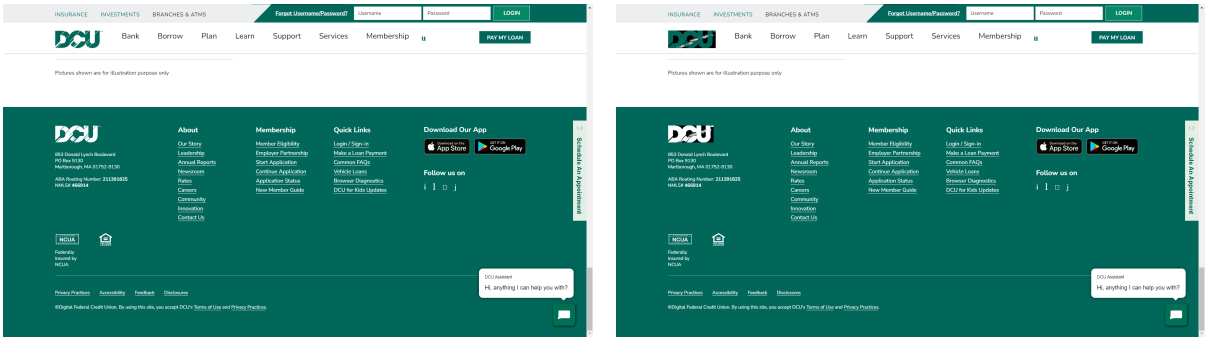
In this section, we discuss a user study to verify whether the phishing webpages generated by PhishOracle truly deceive the actual users. Moreover, we describe our PhishOracle web app, which is designed for both user education and phishing research purposes.

6.1 User Study

In this section, we assess the effectiveness of adversarial phishing webpages generated by PhishOracle in deceiving users. We conduct the study to evaluate the ability of users to determine the legitimacy of a webpage based solely on its visual appearance, without relying on the URL.

Participants: The user study aims to determine whether PhishOracle-generated adversarial phishing webpages can deceive users based on their visual appearance alone, even if detection models can be evaded. To achieve this, we selected 52 participants within the age range of 19 and 26, consisting of academic students and IT professionals. A recent study [48] examined whether phishing webpages that evade the phishing webpage detection models can still deceive users based on visual appearance alone, which closely aligns with our research objective. Furthermore, the study suggests conducting a brief user study by presenting a limited number of phishing webpage samples to the participants for evaluation. This selection allowed us to assess whether IT professionals – who are generally more knowledgeable about phishing threats and frequently encounter suspicious emails – can analyze a webpage's visual appearance to determine its legitimacy. For academic students, the study also served as a form of user education, emphasizing the importance of inspecting webpage visuals for phishing indicators. We adhered to the ethical guidelines [49] to safeguard user information confidentiality.

Research Question: The core research question (RQ) of the study is: *Can a user classify a webpage as 'phishing' or 'legitimate' when presented with a screenshot with its address bar masked?* To answer this, participants were shown webpage screenshots and asked to classify them as either phishing or legitimate. The goal is to assess whether users can distinguish phishing from legitimate webpages based solely on visual elements, without relying on the URL. While we acknowledge that real-world users typically have access to the address bar, prior report [50] has shown that many do not actively use URLs to verify a website's legitimacy. This report found that 25.5% of users clicked on phishing links, and 18% went further to submit their credentials, indicating that a significant portion of users do not carefully inspect URLs before interacting with webpages.



(a) Legitimate webpage **(b)** Generated Phishing webpage

Fig. 8. User Study: (8a) Legitimate and (8b) PhishOracle-generated Webpage Screenshots

Setup: We initiate the experiment by selecting 100 legitimate webpages and generating 100 corresponding adversarial phishing webpages using PhishOracle, by embedding randomly selected visual-based features (refer Table 2). Figure 8a and Figure 8b showcase exemplar screenshots of legitimate and PhishOracle-generated phishing webpages respectively, employed in the user study. The phishing webpage screenshot (Figure 8b) is generated by embedding *V4* visual-based phishing feature into the corresponding legitimate webpage (Figure 8a).

Distribution of Samples: To facilitate this, we utilize Google Forms, offering two options for each webpage screenshot: "Phishing" or "Legitimate". We create 11 Google Forms by randomizing the arrangement of legitimate and PhishOracle-generated adversarial phishing webpage screenshots. Employing Google Forms, we administer the user study to the 52 participants. The first nine forms comprise a mix of 10 legitimate and 15 generated phishing webpage screenshots. One form featured 10 legitimate and 16 generated phishing webpage screenshots, while another included 10 legitimate and 20 generated phishing webpage screenshots. The URL field in each webpage screenshot is masked to ensure unbiased classification outcomes.

Results: Evaluating the feedback of participants reveals that, on average, ~48% of the phishing webpages generated by PhishOracle are misclassified as legitimate. The user study results are presented in Figure 9.

Observations: Our findings from the experiments (refer Section 5.4) and the user study emphasize that the logo transformation techniques (such as opacity, rotation, blur, noise, grey-scale mesh) not only affect the performance of the phishing webpage detection models but also deceive the users. This aligns with the work by Ying *et al.* [51], who argue that assessing the user's perception of adversarial phishing webpages is a necessary step in phishing webpage detection, as users are the actual targets.

Limitations. In our user study, the webpage screenshots do not include URLs, which could assist users in assessing the overall legitimacy of the webpage. Instead, users are presented with webpage screenshots and tasked to classify the webpage as 'phishing' or 'legitimate' based solely on visual appearance.

6.2 PhishOracle Web App

We next introduce our PhishOracle web app, an interactive tool developed to serve both user education and phishing research purposes. This web app facilitates the process of generating phishing webpages by utilizing a legitimate input URL.

The web app was initially hosted on PythonAnywhere[7], a cloud-based platform designed for hosting Python web apps, ensuring accessibility and reliability. However, the hosted web app was disabled due to a violation

[7]PythonAnywhere, https://www.pythonanywhere.com/

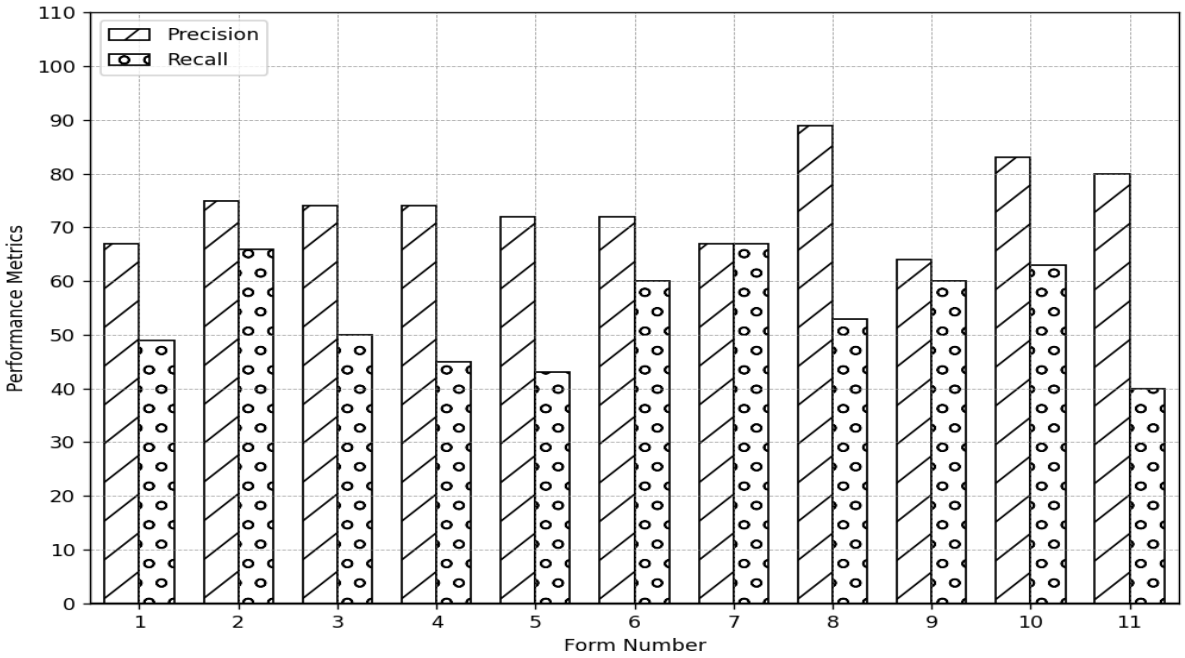
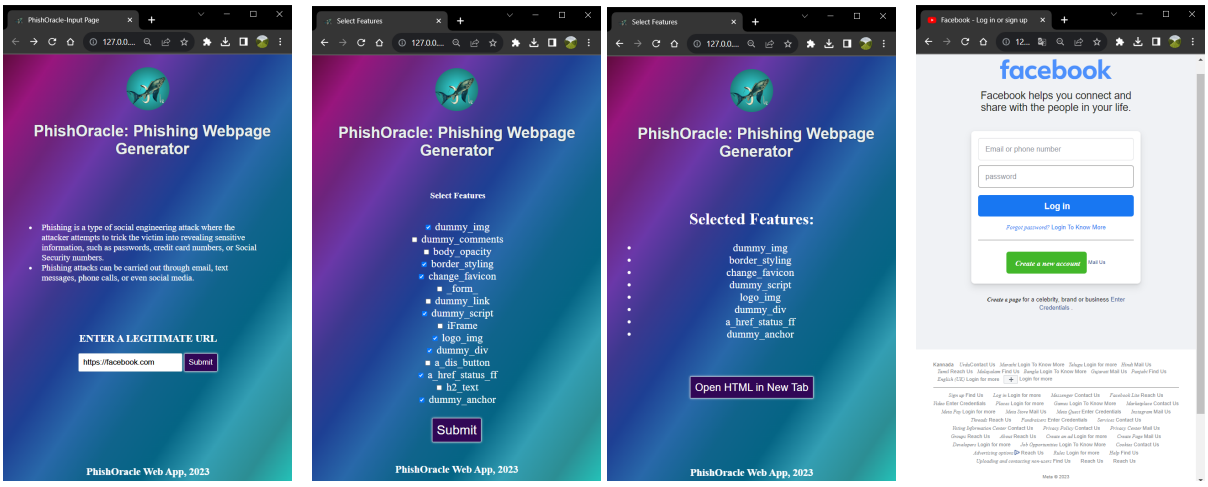


Fig. 9. Results of our User Study



(a) Index Page **(b)** Feature Selection Page **(c)** Generate Page **(d)** Phishing Webpage

Fig. 10. Generating Phishing Webpage via `PhishOracle` Web App

of the Terms and Conditions of PythonAnywhere as the content is related to phishing. We then hosted the

web app on GitHub pages[8], but the generated phishing webpage was marked "Dangerous Site" because of the application containing phishing activities. The web app is thus added as a repository on GitHub [23] which can be downloaded and hosted on a local system. The web app comprises three rendering HTML templates: `index.html`, `select_features.html`, and `generate_webpage.html`. The `index.html` (Figure 10a) serves as the user's entry point, offering an input text box for legitimate URL submission. Once a URL is provided, the application fetches the source code of the associated webpage, extracts relevant features, and presents them as checkboxes on the `select_features.html` page (Figure 10b), allowing users to select features to incorporate into the webpage. After selecting features, the application embeds the chosen features into the legitimate webpage to generate the corresponding phishing webpage, which can be viewed on the `generate_webpage.html` page (Figure 10c). Users can open the generated phishing webpage (Figure 10d) in a new browser tab with a single click. `PhishOracle` web app streamlines the process of generating phishing webpages for research and testing, offering a user-friendly interface for feature selection and webpage generation.

7 LIMITATIONS

While our black-box approach demonstrates the effectiveness of visually transformed adversarial phishing webpages in evading detection models, it comes with some limitations. Our work focuses on a limited subset of transformation techniques and does not consider structural modifications such as background or layout changes. Although on the positive side, the selected transformations are simple to implement, visually deceptive to users, and less complex for real-world adversaries with limited knowledge. Additionally, we evaluated only one LLM-based phishing detection system. Evaluating against a broader set of LLMs is needed to better understand the generalizability of our approach to modern LLM-based phishing webpage detections.

8 CONCLUSION

In this paper, we developed `PhishOracle` to generate adversarial phishing webpages by embedding randomly selected content-based and visual-based phishing features into legitimate webpages. These adversarial phishing webpages were used to evaluate the robustness of the Stack model, VisualPhishNet, Phishpedia, and an LLM-based phishing detector. Our findings reveal that the Stack model exhibits reduced performance, while both VisualPhishNet and Phishpedia incorrectly identify the brands on adversarial phishing webpages due to various logo transformation techniques, leading to incorrect classifications. On the other hand, LLM-based phishing detection using Gemini (*LLM-PD*^{S}) demonstrates better performance against these adversarial attacks, although its detection capability also drops due to `PhishOracle`-generated phishing pages. These results highlight the vulnerabilities of existing phishing webpage detection models, emphasizing the potential of MLLMs in advancing phishing webpage detection. Furthermore, our evaluation of tools from different security vendors on VirusTotal shows that adversarial phishing webpages generated by `PhishOracle` remain undetected within the first 24 hours, leaving them accessible to potential victims for an extended period. Moreover, the user study demonstrates that on an average ∼48% of the `PhishOracle`-generated adversarial phishing webpages are misclassified as legitimate, by the participants. The datasets, `PhishOracle` code and web app are available on our GitHub repositories [22, 23]. `PhishOracle` web app empowers users to clone and host it locally, enabling them to input a legitimate URL, choose specific phishing features, and generate corresponding phishing webpages.

The visual-based features in our tool add opacity to the logos and utilize transformation techniques such as rotation, blurring, adding grey-colored mesh, and noise on logos within a webpage. In our future work, we plan to add a few more techniques to generate adversarial logos and incorporate them as a feature in `PhishOracle` tool.

[8]GitHub Pages, https://pages.github.com/

Ethical Statement. Our institution does not require any formal IRB approval to carry out the research discussed here. Adhering to the principles outlined in the Menlo report [52], our user study is conducted with strict adherence to ethical guidelines, ensuring the non-forging and intentional avoidance of collecting sensitive information from participants. The study involved participants selecting an option ('Phishing' or 'Legitimate') based on the visual appearance of webpage screenshots. Although we plan to release the source code of `PhishOracle` tool on GitHub repositories [22, 23], rather than making it publicly available without restrictions, we will provide access on 'Request Access' basis to ensure responsible use. It is intended solely for educational and research purposes and not for any illegal or unethical activities.

REFERENCES

[1] APWG. Phishing Activity Trends Report, 2024. Available on: https://docs.apwg.org/reports/apwg_trends_report_q1_2024.pdf [Accessed: July 3rd, 2024].

[2] Hossein Shirazi, Bruhadeshwar Bezawada, and Indrakshi Ray. "Kn0w Thy Doma1n Name": Unbiased Phishing Detection Using Domain Name Based Features. In *Proceedings of the 23nd ACM on symposium on access control models and technologies*, 2018.

[3] Ankit Kumar Jain and Brij B Gupta. PHISH-SAFE: URL Features-Based Phishing Detection System Using Machine Learning. In *Cyber Security: Proceedings of CSI 2015*, pages 467–474. Springer, 2018.

[4] Amirreza Niakanlahiji, Bei-Tseng Chu, and Ehab Al-Shaer. PhishMon: A Machine Learning Framework for Detecting Phishing Webpages. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 220–225. IEEE, 2018.

[5] Routhu Srinivasa Rao, Tatti Vaishnavi, and Alwyn Roshan Pais. CatchPhish: Detection of Phishing Websites by Inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing*, 11(2):813–825, 2020.

[6] Anthony Y Fu, Liu Wenyin, and Xiaotie Deng. Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD). *IEEE transactions on dependable and secure computing*, 3(4):301–311, 2006.

[7] 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.

[8] 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, Association for Computing Machinery, 2020. ACM New York, NY.

[9] 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.

[10] 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)*, 2022.

[11] Dinil Mon Divakaran and Sai Teja Peddinti. Large Language Models for Cybersecurity: New Opportunities. *IEEE Security & Privacy*, 2024.

[12] Jehyun Lee, Peiyuan Lim, Bryan Hooi, and Dinil Mon Divakaran. Multimodal Large Language Models for Phishing Webpage Detection and Identification. In *APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2024.

[13] Giovanni Apruzzese and VS Subrahmanian. Mitigating Adversarial Gray-Box Attacks Against Phishing Detectors. *IEEE Transactions on Dependable and Secure Computing*, 20(5):3753–3769, 2022.

[14] 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.

[15] Jehyun Lee, Zhe Xin, Melanie Ng Pei See, Kanav Sabharwal, Giovanni Apruzzese, and Dinil Mon Divakaran. Attacking logo-based phishing website detectors with adversarial perturbations. In *European Symposium on Research in Computer Security*, pages 162–182. Springer, 2023.

[16] Fujiao Ji, Kiho Lee, Hyungjoon Koo, Wenhao You, Euijin Choo, Hyoungshick Kim, and Doowon Kim. Evaluating the Effectiveness and Robustness of Visual Similarity-based Phishing Detection Models. In *34 USENIX Security Symposium*, 2025.

[17] Fabien Charmet, Tomohiro Morikawa, Akira Tanaka, and Takeshi Takahashi. VORTEX: Visual phishing detectiOns aRe Through EXplanations. *ACM Transactions on Internet Technology*, 24(2):1–24, 2024.

[18] Erickson Hyppolite Poel. BlackEye. https://github.com/EricksonAtHome/blackeye [Accessed: July 1st, 2024].

[19] Tahmid Rayat. ZPhisher: Automated Phishing Tool, 2023. https://github.com/htr-tech/zphisher [Accessed: April 10th, 2023].

[20] Abir Hasan. ShellPhish. https://github.com/AbirHasan2005/ShellPhish [Accessed: July 1st, 2024].

[21] 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, Elsevier*, 94:27–39, 2019.

[22] Kulkarni Aditya. PhishOracle-Project, 2025. Available on: https://github.com/LetsBeSecure/PhishOracle-Project.

[23] Kulkarni Aditya. PhishOracle-Webapp, 2025. Available on: https://github.com/LetsBeSecure/PhishOracle-Webapp.
[24] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium, NDSS*, 2019.
[25] Google. Word2Vec, 2013. https://code.google.com/archive/p/word2vec/ [Accessed: June 23th, 2024].
[26] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648, 2007.
[27] Gaurav Varshney, Manoj Misra, and Pradeep K Atrey. A phish detector using lightweight search features. *Computers & Security*, 62:213–228, 2016.
[28] 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*, 2015.
[29] 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*, 2015.
[30] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019. https://github.com/facebookresearch/detectron2.
[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.
[32] Jing Wang, Weiqing Min, Sujuan Hou, Shengnan Ma, Yuanjie Zheng, Haishuai Wang, and Shuqiang Jiang. Logo-2K+: A Large-Scale Logo Dataset for Scalable Logo Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6194–6201, 2020.
[33] Jiawei Han, Micheline Kamber, and Jian Pei. Getting to Know Your Data. In *Data mining*, pages 39–82. Elsevier, 2012.
[34] 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.
[35] Yuexin Li, Chengyu Huang, Shumin Deng, Mei Lin Lock, Tri Cao, Nay Oo, Hoon Wei Lim, and Bryan Hooi. KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 793–810, 2024.
[36] Bushra Sabir, M Ali Babar, Raj Gaire, and Alsharif Abuadbba. Reliability and Robustness analysis of Machine Learning based Phishing URL Detectors. *IEEE Transactions on Dependable and Secure Computing*, 2022.
[37] Dinil Mon Divakaran and Adam Oest. Phishing Detection Leveraging Machine Learning and Deep Learning: A Review. *IEEE Security & Privacy*, 20(5):86–95, 2022.
[38] Qingying Hao, Nirav Diwan, Ying Yuan, Giovanni Apruzzese, Mauro Conti, and Gang Wang. It Doesn't Look Like Anything to Me: Using Diffusion Model to Subvert Visual Phishing Detectors. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3027–3044, 2024.
[39] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014.
[40] 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.
[41] Google. Gemini 1.5 Flash, 2024. Available on: https://ai.google.dev/gemini-api/docs/models/gemini#gemini-1.5-flash [Accessed: February 16th, 2025].
[42] Yun Lin and Ruofan Liu. Phishpedia, 2023. Available on: https://github.com/lindsey98/Phishpedia [Accessed: December 23rd, 2023].
[43] Jehyun Lee, Pingxiao Ye, Ruofan Liu, Dinil Mon Divakaran, and Mun Choon Chan. Building Robust Phishing Detection System: an Empirical Analysis. *NDSS MADWeb*, 2020.
[44] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *Computer Vision–ACCV: 13th Asian Conference on Computer Vision, Springer*, pages 198–213, 2017.
[45] Virustotal, Subsidiary of Google. [n.d.]. Free Online Virus, Malware and URL Scanner. Available on: https://www.virustotal.com/ [Accessed: February 18th, 2025].
[46] Kiho Lee, Kyungchan Lim, Hyoungshick Kim, Yonghwi Kwon, and Doowon Kim. 7 Days Later: Analyzing Phishing-Site Lifespan After Detected. In *THE WEB CONFERENCE*, 2025.
[47] Euijin Choo, Mohamed Nabeel, Doowon Kim, Ravindu De Silva, Ting Yu, and Issa Khalil. A Large Scale Study and Classification of VirusTotal Reports on Phishing and Malware URLs. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(3):1–26, 2023.
[48] Ajka Draganovic, Savino Dambra, Javier Aldana Iuit, Kevin Roundy, and Giovanni Apruzzese. "Do Users Fall for Real Adversarial Phishing?" Investigating the Human Response to Evasive Webpages. In *2023 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–14. IEEE, 2023.
[49] Peter Finn and Markus Jakobsson. Designing Ethical Phishing Experiments. *IEEE Technology and Society Magazine*, 26(1):46–58, 2007.

[50] ID Agent. You'll Be Shocked By The Percentage of Employees Clicking Phishing Emails, 2021. Available on: https://www.idagent.com/blog/youll-be-shocked-by-the-percentage-of-employees-still-clicking-phishing-emails/ [Accessed: February 13th, 2025].
[51] Ying Yuan, Qingying Hao, Giovanni Apruzzese, Mauro Conti, and Gang Wang. "Are Adversarial Phishing Webpages a Threat in Reality?" Understanding the Users' Perception of Adversarial Webpages. In *Proceedings of the ACM on Web Conference 2024*, pages 1712–1723, 2024.
[52] Michael Bailey, David Dittrich, Erin Kenneally, and Doug Maughan. The Menlo Report. *IEEE Security & Privacy*, 10(2):71–75, 2012.