USENIX Security '25 Artifact Appendix: Evaluating the Effectiveness and Robustness of Visual Similarity-based Phishing Detection Models

Fujiao Ji[1], Kiho Lee[1], Hyungjoon Koo[2], Wenhao You[3], Euijin Choo[3], Hyoungshick Kim[2], Doowon Kim[1]

[1]*University of Tennessee, Knoxville* [2]*Sungkyunkwan University* [3]*University of Alberta*

A Artifact Appendix

A.1 Abstract

Phishing attacks pose a significant threat to Internet users, with cybercriminals elaborately replicating the visual appearance of legitimate websites to deceive victims. Visual similarity-based detection systems have emerged as an effective countermeasure, but their effectiveness and robustness in real-world scenarios have been underexplored. In this paper, we comprehensively scrutinize and evaluate the effectiveness and robustness of popular visual similarity-based anti-phishing models. To facilitate reproducibility and accelerate scientific progress–particularly in strengthening collective efforts in combating phishing attacks–we share the resources while adhering to ethical considerations. Given that our work primarily involves evaluation, encompassing multiple datasets and experiments, this artifact appendix highlights key claims and experiments conducted with the Phishpedia detection model to demonstrate the process. Detailed information is available in our GitHub repository https://github.com/Fujiaoji/PhishingEval.

A.2 Description & Requirements

In this section, we provide the essential details needed to replicate the same experimental setup and execute the artifact.

A.2.1 Security, privacy, and ethical concerns

Our research involves the APWG eCX phishing dataset. To maintain ethical standards, we provide HTML, screenshots, and domains without revealing URLs. Furthermore, we utilize open-source models and websites.

Note that, the apwg451514 dataset requires a large amount of space. Additionally, when running the 'DynaPhish' model, please avoid using large datasets, as exceeding the free account credits will result in additional charges.

A.2.2 How to access

We share the resources on our website (https://moa-lab.net/evaluation-visual-similarity-based-phishing-detection-models/) and Zenodo (https://doi.org/10.5281/zenodo.14804193). The code is available on our GitHub repository (https://github.com/Fujiaoji/PhishingEval). Note that, we share the apwg451514 dataset through OneDrive since it occupies a very large space. Please fill out the form on our website to access this share.

A.2.3 Hardware dependencies

We run the codes on both CPU and GPU (NVIDIA A30).

A.2.4 Software dependencies

We run the codes on 'Ubuntu 24.04.1 LTS' and organize the environments through Anaconda. DynaPhish requires the online search and therefore needs to register the Google Cloud account and a browser for execution. Since there are seven baseline models and each model requires a distinct runtime environment, we therefore provide detailed installation instructions a bash file and respective txt file. Due to the page limitations, we share more details in the 'README.md' file on GitHub.

A.2.5 Benchmarks

The artifact mainly contains three components: **(I)** our collected datasets, **(II)** codes, and **(III)** retrained models.

Specifically, the **(I)** collected datasets consist of eight sections: (1) the APWG dataset (apwg451514, Table 2), (2) a sampled subset of the APWG dataset (phishing4190, Table 3), (3) a selected failed 6000 examples from APWG dataset (failed_examples_csv, Table 3), (4) a general benign dataset covering 100 domains (archive100, Table 4), (5) a benign dataset for 110 common brands between phishing datasets and reference lists, also used for ablation studies and manipulations (crawl_benign, Table 7, Table 8, and Table 9), (6) reference lists (expand277, expand277_new, merge277,

merge277_new), (7) a visible manipulation dataset (visible_dataset2, Table 8 and Table 9), and (8) a perturbed dataset (perturbated_dataset, Table 8 and Table 9).

The **(II)** code repository (PhishingEval) contains (1) seven codes of detection models used for evaluating different datasets, (2) crawler.py, a script used for data collection, (3) run_fastdup_cluster.py used for clustering and helping us filter the dataset, and (4) data_test provides three samples for testing.

The **(III)** retrained models are shared in the Zenodo. You can download it for each through the 'download_model.sh' file. Or you can download the folder manually and put them under appropriate positions based on the model structures of the 'README.md' file in the GitHub.

A.3 Set-up

In this section, we outline the installation and configuration steps for Phishpedia, as the setup steps for seven models follow a similar process.

A.3.1 Installation

First, download the repository as a '.zip' file instead of using 'git clone', as the dataset was previously uploaded to GitHub, resulting in extensive history. Follow the total structure outlined in the 'README.md' file to properly download and organize the folders. To run a small sample in the data_test, you need to download at least the corresponding target list datasets through the following command `bash download_data.sh <dataset name>`. For example, Phishpedia requires the logo-based target lists dataset, which can be downloaded through `bash download_data.sh expand277_new`.

After obtaining the code and datasets, the environment can be set up following these steps. Some packages are extracted to mitigate environmental conflicts, making them easier to use and replace.

```
cd PhishingEval/code/reproduce_phishpedia
bash download_model.sh # download models
bash setup_cpu.sh
conda activate env_phishpedia
```

A.3.2 Basic Test

The testing sample information is stored in the 'data_test.csv' file. To use different inputs, please set the parameter of 'input_csv' to your desired CSV path. The new CSV should follow the same column structure as the 'data_test.csv'. Then we can use the following command to run the example. Successful results will be saved in a CSV file. We provide the running code for testing sample and the phishing 4190 dataset in GitHub.

```
conda activate env_phishpedia
python eval_phishpedia.py -
    siamese_weights=models/bit.pth.tar -
    targetlist=../../data/targetlist/
    expand277 -input_csv=../../data/
    data_test/data_test.csv -input_folder
    =../../data/data_test
```

A.4 Evaluation workflow

Our work primarily focuses on evaluation. Reproducing the results requires running the entire set of experiments, which is time-consuming. To facilitate verification, we recommend using the sampled 'phishing4190', as all experiments follow a similar workflow. In this section, we provide some main claims and experiments. Additionally, we demonstrate the time and resource requirements based on the Phishpedia model.

A.4.1 Major Claims

(C1): Reference list-based models can introduce weaknesses. Logos or screenshots not included in the reference list but known to users may mislead the detection models. This highlights the need to constantly expand and update the reference lists and detection models. This is supported by the results in Section 5.1 (E1, E3), as shown in Table 2 and 3.

(C2): Phishpedia demonstrates a better detection rate on the D_{sample} compared to DynaPhish and PhishIntention. Inaccurate keyword extraction can degrade performance, while diverse screenshots and similar web designs present challenges to screenshot-based methods. Selecting an appropriate model structure is crucial to optimizing performance and mitigating these weaknesses. This is also proved by the results in Section 5.1 (E1, E3), as shown in Table 2 and 3.

(C3): Logo-based models currently offer the most reliable approach for standard phishing detection and brand identification, but they are susceptible to additional checking steps, used features, and logo components. Screenshot-based models struggle with web design diversity but may serve as a complementary solution for scenarios involving unknown or emerging brands. This is proved by the brand identification experiments (E2, E3) of Table 2 and 3 in Section 5.3.

(C4): Simple visible and perturbation-based manipulations significantly disrupt logo-based methods. Both are transferable. Screenshot-based methods maintain stable detection but struggle with identifying brands when logos are altered. This is proved by the robustness experiments (E4) of Table 8 and 9 in Section 6.1.

A.4.2 Experiments

After running the code, the results should be saved in a CSV file containing detection and/or image similarity outcomes. Based on these results, you can analyze the detection or identification performance. In this section, we highlight key experiments to provide further insights. The execution and results presented are based on Phishpedia with apwg451514 dataset. When running the code, you can replace the dataset with other smaller-size datasets.

(E1): *[Detection Effectiveness in Section 5.1] [30 human-minute + 42 compute-hour + 250GB disk]:*
Preparation: Download the apwg451514 dataset from Zenodo, prepare the input CSV files, prepare the models based on the set-up section, and ensure the availability of sufficient GPU, CPU, and disk resources.
Execution: Run the evaluation codes.
Results: After getting the detection results, apply predefined thresholds to decide the results. In this step, we assume that images with high similarity to the target will be thought of as phishing, as the entire dataset is phishing.

(E2): *[Phishing Brand Identification in Section 5.3] [60 human-minute + 42 compute-hour + 250GB disk]:* The process is similar to E1. However, after getting the results from models, it is necessary to determine if the predicted brands match the true target brands. Since the same brand may appear under different names, manual verification is required to ensure accuracy.

(E3): *[Detection and Identification Results on D_{sample} in Section 5.] [30 human-minute + 40 compute-minute + 30GB disk]:* Follows the same process as E1 and E2 but with a smaller sample dataset.

(E4): *[Robustness Evaluation in Section 6.1] [120 human-minute + 1 compute-hour + 45GB disk]:* The process contains both detection and identification on the manipulated dataset. Manual verification is also required to ensure the accuracy.

A.5 Notes on Reusability

It is important to make the name of brands consistent when dealing with datasets. Additionally, as original codes for cited papers are frequently updated, it is important to check for the latest versions to maintain compatibility and accuracy.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.