ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX Security ’25 Artifact Appendix: Doubly Dangerous: Evading Phishing Reporting Systems by Leveraging Email Tracking Techniques

Anish Chand
Louisiana State University

Nick Nikiforakis
Stony Brook University

Phani Vadrevu
Louisiana State University

A Artifact Appendix

A.1 Abstract

In our paper, we repurposed various email tracking vectors to profile and evade the anti-phishing systems employed by major email service providers. We identified the different HTTP headers and tracking vectors triggered by the email services during different stages like email prefetching, email opening and after email reporting. Due to the distinct behavior exhibited by the email systems, we were able to identify and evade them using evasive phishing attacks. The artifact provided contains the dataset and the code used for the project. The dataset is accompanied by a helper script to execute queries to generate the results presented in the paper. README files are also included to facilitate easy navigation and reproduction of results found in the paper.

A.2 Description & Requirements

To support these findings, this artifact is provided in two main components:

1. (Dataset and Results): We provide the dataset in a self-contained Docker environment that includes a PostgreSQL database injected with the dataset from our experiments. We also provide an accompanying Python script that allows for the reproducibility of the major tables and figures presented in our paper.
2. Part B (Code): The code provided includes three software components: email-sending module, a websites module and a GUI automation module. This part uses Docker to run the database but runs our three main applications (two Laravel web apps, one Python automation script) locally on host machine. These applications run locally with a small, artificial database to demonstrate their architecture and testability without requiring live credentials or external network access such as live web servers and domains.

A.2.1 Security, privacy, and ethical concerns

Throughout our research, we took several measures to mitigate any security, privacy or ethical concerns. Our investigation findings pose security risks to the email providers if they are exploited and is not intended for unauthorized or malicious actors. Hence, we have made the dataset and the websites code module that exploits the dataset to launch evasive phishing campaigns are available with restricted access on Zenodo. Additionally, the provided applications are operating locally without the ability to perform live actions such as sending emails and launching phishing sites. There are no destructive steps, and no security mechanisms on the evaluator’s machine need to be disabled.

A.2.2 How to access

Our artifact is available on Zenodo. Our public version with the email sending code and the GUI automation code are available here: https://doi.org/10.5281/zenodo.15612284. Our restricted repository that contains the dataset, websites module code, email sending code and GUI automation code are available here: https://doi.org/10.5281/zenodo.15612193

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

The following tools are required to be installed on your device:

- **Operating System:** Linux (tested on Ubuntu 22.04) or macOS (tested on macOS 15.5). The GUI automation script has only been tested on Windows 11.
- **Docker and Docker Compose:** For running the PostgreSQL database containers.
- **PHP 8.3+** and **Composer 2.x**: For running the Laravel applications locally in Part B.
- **Python 3.9+** and **virtualenv**: For running the analysis and automation scripts.
- **The ‘whois’ command-line utility:** This is a mandatory dependency for Part B (the Functional Demo). Our Laravel application uses it to perform lookups on IP addresses.

A.2.5 Benchmarks

None.

A.3 Set-up

This section provides the complete installation and configuration steps for the artifact.

A.3.1 Installation

This setup allows for the direct regeneration of our paper's results from the provided dataset.

1. **Start the Environment:** Navigate to the `results` directory. Then, execute the following command to start the PostgreSQL container and automatically load the full dataset:

   ```
   docker-compose up --build -d
   ```

2. **Install Python Dependencies:** Navigate to the `results/analysis` directory and run:

   ```
   pip install -r requirements.txt
   ```

3. **Start the Database:** Navigate to the `code` directory and start the PostgreSQL container:

   ```
   docker-compose up -d
   ```

4. **Set up the email-sender and database:** Go to the application directory `email-sending`, install dependencies and configure the application:

   ```
   composer install
   cp .env.example .env
   php artisan key:generate
   php artisan migrate --seed
   ```

   Expected Output: The console will show successful migration and seeding messages. The database is now correctly structured and populated for the demo.

5. **Run the websites Application:** This application serves the web pages used for phishing simulation and honey sites. Navigate to its directory, `websites-code`, install its dependencies and then start the local web server:

   ```
   composer install
   cp .env.example .env
   php artisan key:generate
   php artisan serve --port=8000
   ```

A.3.2 Basic Test

1. **Test for dataset evaluation:** To verify that the analysis script is functional in *results/analysis*, run its help command:

   ```
   python queries.py -h
   ```

   Expected Output: The script will print help and options to the console.

2. **Test for websites-code:** Open a web browser and navigate to `http://localhost:8000` after `php artisan serve -port=8000`.

   Expected Output: The application's homepage should load successfully in the browser, confirming that the web server is running and connected to the demo database.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *The major email providers namely Gmail, Outlook, and Proton, all employ email subsystems that can be identified using a combination of unique HTTP headers and tracking vectors. Tables 7, 8, and 9 showing unique HTTP headers and Tables 4, 5, and 6 showing the unique tracking vectors triggered highlight this.*

(C2): *The tracking vectors and HTTP headers can be used to launch evasive phishing sites that can avoid detection from the email anti-phishing entities compared to the baseline normal sites.*

A.4.2 Experiments

(E1): *There are no experiments that can be run. However, all of our major claims can be validated with the provided queries.py script. The instructions to setup are in the README file in the folder and the environment can be set up in a matter of some minutes. The script will output the tables in the console and may output figure when applicable.*

(E2): *Similar to above, this claim can be validated with the provided queries.py script that will output the number of email reports made for the smart site, which is significantly higher compared to the baseline sites. The provided websites-code can be used to evaluate the evasion logic used compared to the baseline sites as well as the phishing payload used. The accompanying README in the folder provides additional steps that may be helpful in evaluation of the code.*

A.5 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/.