

PhishAware: A Web-Based Phishing Simulation and Awareness Training System

Sapna Pitchamani*, Maya Nair**

*(Department of Computer Science, SIES College of Arts, Science and Commerce, Mumbai, India
Email: nadarsapna66@gmail.com)

** (Department of Computer Science, SIES College of Arts, Science and Commerce, Mumbai, India
Email: mayan@sies.edu.in)

Abstract:

Phishing succeeds by exploiting people rather than software, yet many awareness programmes rely on generic training that tells an organisation little about how its own users behave. This paper presents PhishAware, a web-based phishing simulation and awareness training system built with Python and Flask. An administrator creates a campaign through a guided form, chooses one of eight pretext templates or writes a custom email, and sends it to a list of recipients over SMTP. Each campaign carries a unique link; when it is opened, the click is logged with a timestamp and the user sees an awareness page explaining how to recognise suspicious messages. A dashboard and a reports page summarise campaigns, emails sent, recorded clicks and training completions. The system never asks for or stores credentials. A local test with three test mailboxes shows that the sending, tracking, training and reporting pipeline works end to end. Because the test group is very small and tracking is per campaign, the results are a functional validation and not evidence of behavioural change. Limitations and the work needed for a real user study are discussed.

Keywords — phishing simulation, security awareness training, click tracking, Flask, web application.

I. INTRODUCTION

Phishing remains one of the most common ways for attackers to reach accounts and data, because it targets the person reading a message instead of a flaw in software. Small and medium-sized organisations are especially exposed. They often have no dedicated security staff, and their awareness training tends to be a slide deck or an annual video rather than practice with realistic messages. Commercial simulation platforms exist, but they are priced and scoped for larger organisations, and published research systems that resemble them are still few [1].

It is also less settled than vendor reports suggest whether simulation and training change behaviour at all. A randomised trial of nearly 20,000 hospital employees found little benefit from annual training and only a small effect from training shown right after a failed simulation [2]. That result argues for

platforms that record what users actually do and that are honest about what the records can show.

This project builds a small, transparent platform with that aim. Its objectives are to let an administrator run simulated phishing campaigns in a controlled and ethical way, to record how recipients interact with them, to deliver awareness content at the moment of a click, and to summarise the activity in a dashboard. The contributions are as follows:

- A Flask application that lets an administrator build and launch simulated phishing campaigns from a guided form, using one of eight templates or a custom email.
- A click-tracking and training-completion mechanism, with a dashboard and a report page that summarise the logged activity.
- A description of the design, including its ethical safeguards, and a frank account of what the prototype test does and does not demonstrate.

The rest of the paper is organised as follows. Section II reviews related work, Section III presents

the design and methodology, Section IV describes the implementation, Section V reports the testing and results, Section VI discusses limitations, and Section VII concludes.

II. RELATED WORK

A. Simulation Platforms and Embedded Training

Alsaqer et al. [1] describe a customisable platform that delivers simulations by email, SMS and WhatsApp, adapts messages to a user's role and shows interactions on a dashboard. For ethical reasons they could not test it with real users and relied on simulated data. Pairing a simulated attack with immediate feedback is an older idea: Jansson and von Solms [3] ran simulations with embedded training across a university's email users and saw insecure reactions fall between two weekly rounds.

More recent work is less optimistic about why such programmes work. Lain et al. [4] separated the content of the training from the reminder effect of being tested and concluded that most of the benefit comes from the nudge, because employees rarely read the material. Schiller et al. [5] found that simulations can give employees a false sense of security and that only a minority of those who clicked used the e-learning on offer. Small studies such as the age-group comparison of Mohamed Ali and Rahman [6] show that a modest simulation website can still yield interpretable results.

B. Susceptibility

Field and laboratory studies ask what makes people click. Diaz et al. [7] sent three different pretexts to 1,350 students and found that susceptibility varied with the pretext. Moody et al. [8] linked susceptibility to curiosity, risk propensity and the apparent familiarity of the sender. Kleitman et al. [9] found that how malicious a user judges an email to be, together with knowledge and confidence, explained about half of the variance in susceptibility, and Ge et al. [10] showed that the care a user takes in processing a message mediates the effect of personal characteristics. These findings motivate offering several pretexts and recording more than a single yes/no outcome.

C. Measuring the Effect of Training

Ho et al. [2] reported no significant relation between completing annual training and later failure, and a small effect of embedded training. Steves et al. [11] propose the Phish Scale, which rates how hard a simulated email is to detect from its observable cues and from how well its premise fits the target, so that click rates on easy and hard emails are not compared as if they were equal.

D. Language Models and Gamification

Large language models can generate exercise scenarios [12] and phishing text. Heiding et al. [13] compared generic, GPT-4 and persuasion-framework emails sent to 112 participants and reported click-through rates of 19–28% for the control group and up to 79% for the framework-based emails. On the defensive side, ChatSpamDetector [14] uses a language model to classify an email and to explain its decision. A review of gamified awareness training [15] finds short-term gains but little evidence on which game elements matter or whether the effect lasts. PhishAware currently uses fixed templates and a plain awareness page; these results point to difficulty-rated, generated templates and richer training as future work.

Taken together, the literature offers capable platforms and careful studies, but few small, low-cost and fully documented systems that link simulation, tracking, training and reporting and that state plainly what their data can support. PhishAware is a step in that direction.

III. SYSTEM DESIGN AND METHODOLOGY

A. Architecture

The system has four parts, shown in Fig. 1: a browser front end rendered from Jinja templates with HTML, CSS and JavaScript, a Flask application that holds the routes and the business logic, an SMTP connection used to deliver the simulated emails, and a file-based data store. Flask [16] was chosen because it is small and lets the whole flow be read in a single file.

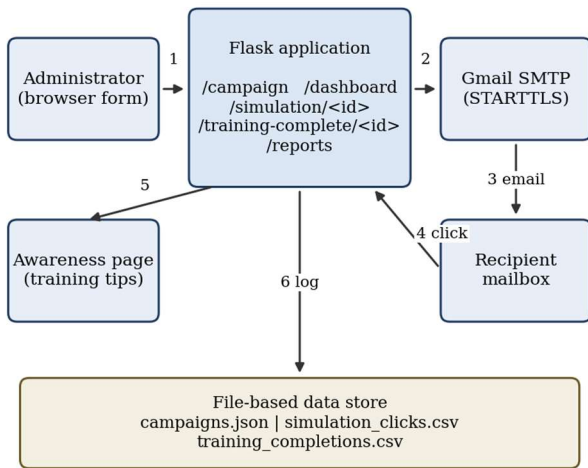


Fig. 1 Architecture and message flow of PhishAware. Numbers show the order of one campaign.

B. Campaign Workflow

An administrator works through a multi-step form that collects the campaign name and description, the recipient addresses, the email content, and a training title and message, and ends with a review step before launch. On launch the server validates the input, creates a campaign identifier, builds the email, sends it to each recipient, and saves the campaign record. If sending fails part-way, the campaign is still saved with a failure status, the error text and the number of messages that were sent successfully.

C. Email Templates

Eight built-in templates cover common pretexts, as listed in Table I. Each has a sender label, a subject and an HTML body in which the call-to-action is a styled button, so the recipient sees a button and not a raw URL. The administrator can instead write a custom email, insert the link with one click, and preview the result before sending.

TABLE I
BUILT-IN EMAIL TEMPLATES

#	Sender Label	Pretext
1	Talent Acquisition	Application shortlisted
2	Account Security	Password reset request
3	Prize Notification	Prize or reward notice
4	Delivery Services	Action needed for a delivery
5	Security Team	Security notice
6	Billing Department	New invoice available
7	Calendar Services	Meeting invitation
8	HR Benefits	Employee benefits update

Each template carries a placeholder that the server replaces with the campaign's tracked link.

D. Link Generation and Click Tracking

Each campaign receives a random version-4 UUID. The server replaces the link placeholder in the email body with an address of the form /simulation/<id>. When that address is opened, the server appends the identifier and a timestamp to a click log, unless the identifier is already present, and then returns the awareness page. Recording only the first click keeps the count to one event per campaign.

E. Awareness Page and Training Completion

The awareness page tells the user that the message was a simulation and gives four short tips: check the sender, inspect links before opening them, watch for urgency and unexpected prizes, and never share passwords through unexpected links. A separate POST route records that a campaign's training was completed, again once per campaign. The page contains no input fields, so nothing the user types can be captured.

F. Data Storage and Dashboard Metrics

Three human-readable files hold all data (Table II). Flat files keep the prototype easy to inspect and need no database server. The dashboard reads them on every request and reports four totals: campaigns, emails sent (the sum of each campaign's sent counter), links clicked (rows in the click log) and training completed (rows in the completion log). For each campaign it also shows the click count and whether training was completed.

TABLE II
DATA FILES AND FIELDS

File	Fields
campaigns.json	id, name, recipients, subject, body, created_at, status, emails sent
simulation_clicks.csv	simulation_id, clicked at
training_completions.csv	simulation_id, completed at

G. Ethical Safeguards

The system is meant only for authorised awareness exercises. It sends only to addresses the administrator enters, it has no page that collects credentials, mail credentials are read from an environment file and not stored in the code, and every click ends on a page that explains the exercise.

During testing the recipients were limited to three test mailboxes.

IV. IMPLEMENTATION

The application is written in Python 3 with Flask. Email is sent with the standard smtplib module through Gmail's SMTP service on port 587 with STARTTLS, using the python-dotenv package to load the account details. The message is multipart: a short plain-text part for clients that cannot show HTML and an HTML part carrying the template. Campaign data is saved with the json module and the logs with the csv module. The routes are listed in Table III.

Input handling is deliberately simple. The campaign name, recipients, subject and body are all required; recipient addresses are split on commas, lower-cased and de-duplicated, and each must contain an at sign. The templates are held in the page's JavaScript, which copies the chosen subject and body into hidden fields that the form submits.

TABLE III
APPLICATION ROUTES

Route	Method	Purpose
/	GET	Home page
/dashboard	GET	Totals and campaign table
/reports	GET	Click totals and click log
/campaign	GET, POST	Create and send a campaign
/simulation/<id>	GET	Log a click, show awareness page
/training-complete/<id>	POST	Log training completion

V. TESTING AND RESULTS

A. Test Setup

The prototype was run on the Flask development server of a single computer, with Gmail SMTP for delivery and three test mailboxes as recipients. Because the tracked link points to the local address 127.0.0.1, a click can be recorded only when the link is opened on that same computer. The test was therefore a check that each part of the pipeline works, not a study of user behaviour.

B. Observed Activity

Table IV gives the totals shown on the dashboard on 6 October 2026. By 8 October the campaign log

held 23 entries. Of those, six contain the tracked address; the others, created during early testing, link to a static awareness page that opens without logging a click. Campaigns created before the sent counter was added are counted as zero emails sent.

TABLE IV
DASHBOARD TOTALS, 6 OCTOBER 2026

Measure	Value
Campaigns created	20
Emails sent	19
Click records in log	7
Training completions in log	3
Distinct recipient mailboxes	3

C. Functional Verification

Table V relates each function to the evidence available. The dashboard showed click counts and a completed-training flag against individual campaigns, which confirms that a click on a tracked link was stored against the right campaign and that the awareness step was recorded for it. Handling of repeated clicks and of a failed send is implemented in the code but was not tested separately, and the table says so.

TABLE V
FUNCTIONS AND EVIDENCE

Function	Evidence
Campaign creation and storage	Campaign log holds 23 records
Email delivery over SMTP	Sent counter recorded per campaign
Click logging	Click counts shown per campaign
Training completion	Completion flag shown per campaign
Dashboard totals	Totals shown on the dashboard
Repeat-click handling	In code; not tested separately
Failed-send handling	In code; not tested separately

D. Discussion

The counts in Table IV show that the pipeline runs from campaign creation to sent email, logged click and recorded training. They are not rates. Clicks are tracked per campaign and only the first is stored, several test campaigns used an untracked link, and all emails went to three mailboxes. No click rate, no comparison between templates and no claim about learning is therefore made. The literature points to what a real evaluation needs: separate links for each

recipient, a difficulty rating for each template [11], enough participants, and effect sizes reported plainly, as in [2].

VI. LIMITATIONS AND THREATS TO VALIDITY

Scale of testing. Three mailboxes and a handful of clicks cannot support statistical claims, and the recipients are not independent participants.

Tracking. One link is shared by all recipients of a campaign and only the first click is stored, so activity cannot be attributed to a person. Early campaigns used a static link and logged nothing.

Deployment. Links point to a local address, so recipients on other devices cannot open them. A public address, for example through a tunnelling service, is needed for a real run.

Storage and access. Flat files are not safe for concurrent users, and the application has no login, so it is unsuitable for production. A database and authenticated administrator access were planned but are not implemented.

Unused inputs. The training title and message fields on the form are collected but not yet used by the server, and the awareness page is the same for every campaign.

Delivery. Sending through a personal mail account is rate-limited and may be filtered as spam; delivery rates were not measured. No performance or usability evaluation was carried out.

VII. CONCLUSION AND FUTURE WORK

This paper presented PhishAware, a Flask-based system that sends simulated phishing campaigns, logs clicks, shows an awareness page and summarises activity in a dashboard, with safeguards that keep it from collecting credentials. A local test with three mailboxes confirmed that the pipeline works from end to end. The main outcome is a documented, working prototype and an honest statement of its limits; the activity counts describe the test and not user behaviour.

Future work will give each recipient a separate token and log opens, clicks and training events per person; add a database and authenticated administrator access; host the application at a public address; use the training fields to deliver template-

specific lessons; rate each template for difficulty; and run a consent-based pilot with a larger group, with a debrief and with results reported as effect sizes. Risk scoring from interaction history and the use of language models to generate difficulty-rated templates, with safeguards, are further extensions.

ACKNOWLEDGMENT

The author thanks Maya Nair and the Department of Computer Science, SIES College of Arts, Science and Commerce, for guidance and the facilities used in this work.

REFERENCES

- [1] A. Alsaqer, H. Almajed, K. Alarfaj, and M. Frikha, "Phishing simulation as a proactive defense: A customizable platform for training and behavioral analysis," *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 6, 2025.
- [2] G. Ho *et al.*, "Understanding the efficacy of phishing training in practice," in *Proc. IEEE Symp. Security and Privacy (SP)*, 2025, doi: 10.1109/SP61157.2025.00076.
- [3] K. Jansson and R. von Solms, "Phishing for phishing awareness," *Behav. Inf. Technol.*, vol. 32, no. 6, pp. 584–593, 2013.
- [4] D. Lain, T. Jost, S. Matetic, K. Kostianen, and S. Capkun, "Content, nudges and incentives: A study on the effectiveness and perception of embedded phishing training," arXiv:2409.01378, 2024.
- [5] K. Schiller, F. Adamsky, C. Eichenmüller, M. Reimert, and Z. Benenson, "Employees' attitudes towards phishing simulations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2024, pp. 4167–4181, doi: 10.1145/3658644.3690212.
- [6] A. N. Mohamed Ali and S. H. Rahman, "Analyzing cybersecurity risk with a phishing simulation website," in *Proc. 9th Int. Conf. Cloud Comput. Internet Things (CCIOT)*, Hanoi, 2024.
- [7] A. Diaz, A. T. Sherman, and A. Joshi, "Phishing in an academic community: A study of user susceptibility and behavior," arXiv:1811.06078, 2018.
- [8] G. D. Moody, D. F. Galletta, and B. K. Dunn, "Which phish get caught? An exploratory study of individuals' susceptibility to phishing," *Eur. J. Inf. Syst.*, vol. 26, no. 6, pp. 564–584, 2017.
- [9] S. Kleitman, M. K. H. Law, and J. Kay, "It's the deceiver and the receiver: Individual differences in phishing susceptibility and false positives with item profiling," *PLOS ONE*, vol. 13, no. 10, p. e0205089, 2018.
- [10] Y. Ge, L. Lu, X. Cui, Z. Chen, and W. Qu, "How personal characteristics impact phishing susceptibility: The mediating role of mail processing," *Appl. Ergon.*, vol. 97, p. 103526, 2021.
- [11] M. P. Steves, K. K. Greene, and M. F. Theofanos, "Categorizing human phishing difficulty: A Phish Scale," *J. Cybersecurity*, vol. 6, no. 1, 2020.
- [12] M. M. Yamin, E. Hashmi, M. Ullah, and B. Katt, "Applications of LLMs for generating cyber security exercise scenarios," *IEEE Access*, vol. 12, pp. 143806–143822, 2024.
- [13] F. Heiding, B. Schneier, A. Vishwanath, J. Bernstein, and P. S. Park, "Devising and detecting phishing emails using large language models," *IEEE Access*, vol. 12, pp. 42131–42146, 2024.
- [14] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, "ChatSpamDetector: Leveraging large language models for effective phishing email detection," in *Proc. 20th EAI Int. Conf. Security and Privacy in Communication Networks (SecureComm)*, 2024.

- [15] A. K. Gwenthure and F. S. Rahayu, "Gamification of cybersecurity awareness for non-IT professionals: A systematic literature review," *Int. J. Serious Games*, vol. 11, no. 1, 2024.
- [16] Pallets, "Flask documentation." [Online]. Available: <https://flask.palletsprojects.com/>