An AI-Assisted Multi-Signal Framework for Phishing Detection, Risk Scoring, and Proactive Protection

Preethi S.[1], Raghav G.[2], John Ashley D.[3], Dev Harshith S.[4], Dhiya Shree S.[5], Dhanyalakshmi S.[6]
[1,2,3,4,5,6]*Department of Computer Science and Engineering, SNS College of Technology, Coimbatore, Tamil Nadu, India*

***Abstract*—Phishing attacks exploit trust through deceptive messages, malicious URLs, lookalike domains, and imitation login pages, and rule- and signature-based controls can be less effective against previously unseen or modified attacks. This paper presents PhishX, an AI-assisted phishing-defence framework that combines message-content analysis, URL and domain features, sender-related signals, risk scoring, and protective actions in one workflow. The proposed pipeline comprises input handling, feature extraction, machine-learning and natural-language-processing analysis, classification into Safe, Suspicious, or Phishing, and a corresponding action: allowing content, warning the user, or quarantining high-risk content and alerting a security team. The described prototype uses a React and Tailwind CSS interface, a Python FastAPI backend, Scikit-learn-based text classification, PostgreSQL storage, and JSON Web Token (JWT) authentication with role-based access. Contributions are a multi-signal analysis design, a configurable weighted risk-scoring model linked to protective actions, a layered architecture with a feedback pathway, and a reproducible evaluation protocol. The project report describes functional test scenarios but provides no benchmark datasets, measured accuracy, false-positive rates, or controlled performance results; accordingly, no quantified detection effectiveness is claimed, and evaluation on labelled datasets with reproducible security and usability tests remains required.**

***Index Terms*—artificial intelligence, machine learning, phishing detection, risk scoring, URL analysis**

I. INTRODUCTION

A. Background

Phishing is a social-engineering attack in which an attacker impersonates a trusted person, organization, or service to persuade a victim to disclose credentials or other sensitive information. Messages may use urgent language, misleading links, lookalike domains, copied login pages, or QR codes to encourage unsafe actions. Increasingly polished messages can also reduce the usefulness of spelling and grammar as warning signs.

B. Technical problem

Conventional filtering commonly relies on known malicious URLs, sender indicators, and fixed patterns [1]. These mechanisms remain useful, but a newly registered domain or a small change in wording may avoid a known rule.

Single-signal detectors also have characteristic blind spots: a text classifier may miss a message with little incriminating wording but a malicious link, whereas a URL classifier may miss a credential-harvesting message that points to a legitimate but compromised site. Many tools also stop at a label and leave the response to the user.

C. Approach and contributions

PhishX is designed to assess several signals together and to connect detection with prevention and response. Rather than returning only a label, it maps the classification to an action intended to reduce the chance that users interact with harmful content.

The paper makes five bounded contributions: (i) a multi-signal analysis design that combines TF-IDF text classification, URL lexical features, and a sender-behaviour score; (ii) a configurable weighted risk-scoring model with three categories and matching actions; (iii) a layered architecture with JWT authentication and role-based access control; (iv) a feedback pathway for reviewed reports; and (v) a reproducible evaluation protocol with explicit reporting requirements.

D. Scope and organization

PhishX is a project prototype; no claim is made that it has been independently validated in production, and the paper does not propose a new classification algorithm. Section II reviews related work; Sections III and IV present the system and its architecture; Section V defines the evaluation approach; and Sections VI and VII discuss limitations and conclude.

II. RELATED WORK

A. Email and URL-based phishing detection

Fette, Sadeh, and Tomasic studied machine-learning approaches to detecting phishing emails using message and link characteristics [2]. Abu-Nimeh et al. compared machine-learning techniques for phishing detection [3]. Sahingoz et al. investigated machine-learning-based phishing detection from URLs [4], while Sahoo, Liu, and Hoi surveyed malicious-URL detection methods [1]. Jain and Gupta examined client-side phishing website detection using machine learning [5]. Together, these works motivate using multiple observable features rather than depending exclusively on a static blacklist.

B. Text representation and supporting technologies

Term-frequency/inverse-document-frequency (TF-IDF) weighting is a long-established way of representing text for retrieval and classification [6], and Scikit-learn provides the machine-learning tools used by the proposed text pipeline [7]. JSON Web Token (JWT) is a compact format for representing claims transferred between parties [8], and role-based access control (RBAC) assigns permissions to roles instead of individual users [9]. FastAPI, React, Tailwind CSS, and PostgreSQL provide the API, interface, styling, and data-persistence technologies described for PhishX [10] [13].

C. Adjacent applied-AI examples

Mubaraali et al. applied a convolutional neural network to IoT-based induction-motor diagnosis [14], and Raj et al. described an AI-enabled system for dimensional measurement and material transport [15]. Neither addresses phishing; they are cited only as general examples of machine learning inside an operational monitoring workflow, not as evidence of phishing-detection performance.

D. Positioning

The contribution claimed here is an integrated workflow combining multiple signals with risk-based actions, access control, and a feedback loop, not a new classifier. No novelty is claimed relative to every existing anti-phishing product.

III. PROPOSED SYSTEM

A. Workflow

PhishX accepts an email/message text or URL for analysis (Fig. 1). The system validates the input, collects available attributes, extracts features, applies text and URL analysis, combines the resulting signals into a risk score, assigns a threat category, and selects a protective action. Empty, malformed, or over-length inputs are rejected with an error instead of being scored. Feedback from reviewed reports is intended to inform later model updates.

B. multi-signal analysis

Text analysis. Message text is converted to TF-IDF features [6] and classified with logistic regression [7]. The output is normalized to a text risk T between 0 and 1, for example the predicted phishing-class probability. The cues a model learns, such as urgency or requests for credentials, depend entirely on the training corpus, which the project report does not describe.

URL analysis. Seven lexical and structural characteristics are extracted (Table I) and combined into a URL risk U between 0 and 1. The exact mapping from features to U is implementation-specific and must be documented and calibrated.

No single feature is conclusive: a brand term in a host is not by itself proof of phishing, and HTTPS does not establish that a site is trustworthy.

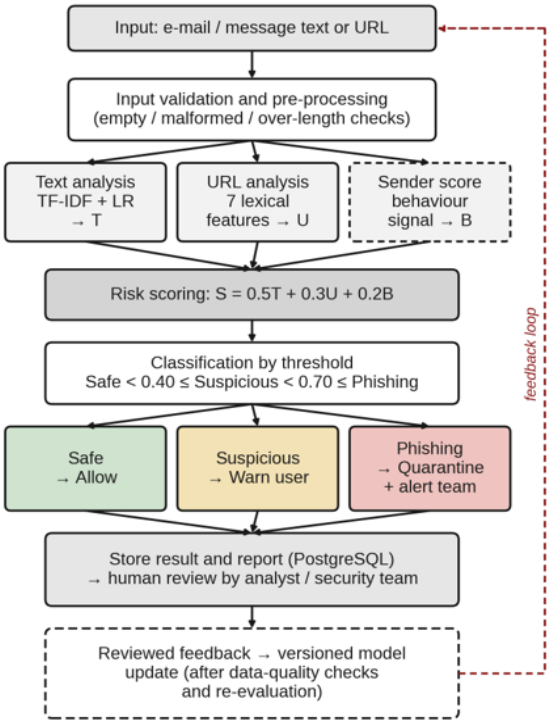


Fig. 1. End-to-end PhishX workflow from input to protective action, with the reviewed-feedback loop.

Table I. URL features extracted by the prototype, with the caveat for each.

Feature	What is measured; caveat
URL length	Total characters; long legitimate links exist
IP-address host	Host is a raw IP; rare, but used in internal tools
Subdomain count	Labels before the domain; CDNs use many
At-sign (@)	Presence of @; can hide the real destination
HTTPS usage	Scheme is https; not proof of trust
Hyphen count	Hyphens in the host; common in lookalike hosts
Brand term	Possible brand name in the host; not proof alone

Sender behaviour. A sender-behaviour score B is included as a further input. The project report does not specify a complete sender-feature extraction method, so candidate indicators beyond the documented score are treated as future work and not as part of the design.

C. Risk scoring

The documented example combines the three signals using configurable weights:

$$S = 0.5T + 0.3U + 0.2B \quad (1)$$

where T is text risk, U is URL risk, and B is the sender-related score. The weights sum to one and each input lies in [0, 1], so S also lies in [0, 1]. Scores below 0.40 are labelled Safe, scores from 0.40 to below 0.70 Suspicious, and scores of 0.70 or above Phishing (Fig. 2). These are prototype rules that should be calibrated against representative labelled data before operational use.

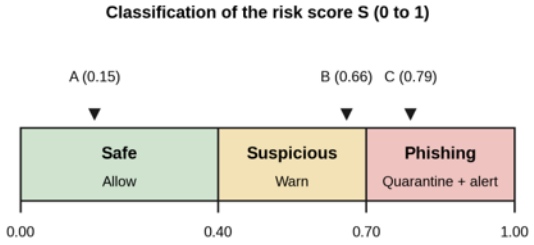


Fig. 2. Threshold-based classification of the risk score S, with the hypothetical cases A–C of Table II.

Table II applies Eq. (1) to three hypothetical inputs to show the arithmetic; the values are illustrative and are not measurements. Raising the Phishing threshold reduces unnecessary quarantine but let's more borderline attacks through as warnings; lowering it does the reverse. Since quarantine is more disruptive than a warning, thresholds should be chosen from the validation precision–recall trade-off [16] rather than fixed in advance. Scores from different models are not automatically on comparable scales, so probability calibration is advisable before weighted fusion.

Table II Illustrative application of Eq. (1) to hypothetical inputs (not measured results).

Case	T	U	B	S	Category
A	0.20	0.10	0.10	0.15	Safe
B	0.80	0.60	0.40	0.66	Suspicious
C	0.90	0.80	0.50	0.79	Phishing

D. Protective response and feedback

Safe content is allowed; Suspicious content triggers a warning; and Phishing content is intended to be quarantined with an alert to the security team. This action mapping makes the system prevention-oriented rather than a classifier-only tool. Actual blocking, quarantine, and alert delivery must be verified in an integrated deployment. Reviewed reports and user feedback are stored so that later model updates can learn from them; feedback should pass review and re-

evaluation before any retraining is released, to keep incorrect reports from degrading the model.

IV. SYSTEM ARCHITECTURE AND IMPLEMENTATION

A. Layered architecture

The architecture separates the user interface, API, analysis logic, and persistence (Fig. 3). Layer 1 is the React and Tailwind CSS frontend, which provides the scanner and dashboard and is not authoritative for permissions. Layer 2 is the FastAPI backend, which accepts scan requests, validates inputs, verifies JWT tokens and roles, invokes the analysis pipeline, and returns a risk score, category, and action. Layer 3 holds the analysis services built on Scikit-learn. Layer 4 is PostgreSQL, identified as the database for user information, analysis results, reports, and feedback. A fifth observability and governance layer is recommended for production use.

B. Scan request sequence

Fig. 4 shows the authenticated scan. The client sends the input with a bearer token; the backend verifies the token, checks the caller's role, and validates the payload before analysis. The pipeline returns the component scores, the backend stores the result, and the client receives the score, category, and action. For Phishing results the quarantine and alert path is triggered, and a separate feedback endpoint lets users and reviewers submit outcomes for later improvement.

C. Data model and access control

The core records are users with roles, scan requests, analysis results (T, U, B, S, category, action, and model version), reports, and feedback. The project report states that role-based access is intended to restrict protected operations [9] but does not enumerate roles. A least-privilege set is therefore proposed for evaluation: a user may scan inputs, view own results, and submit feedback; an analyst may review quarantined items, resolve alerts, and label feedback; and an administrator may manage users, weights, thresholds, and model releases, with every such change logged.

D. Prototype scope

The prototype code in the project report demonstrates URL feature extraction, weighted risk combination, threshold-based classification, an authenticated scan endpoint, a feedback endpoint, and a React scanner component. The code is illustrative of the intended design; its presence does not establish secure deployment, reliable quarantine, or measured detection performance. In a production implementation, secrets must be loaded from protected configuration, inputs must be validated, and access checks must be enforced server-side.

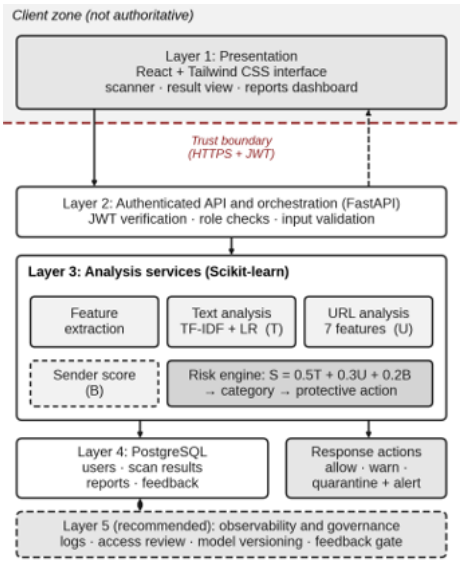


Fig. 3. Layered architecture of PhishX showing the client–server trust boundary.

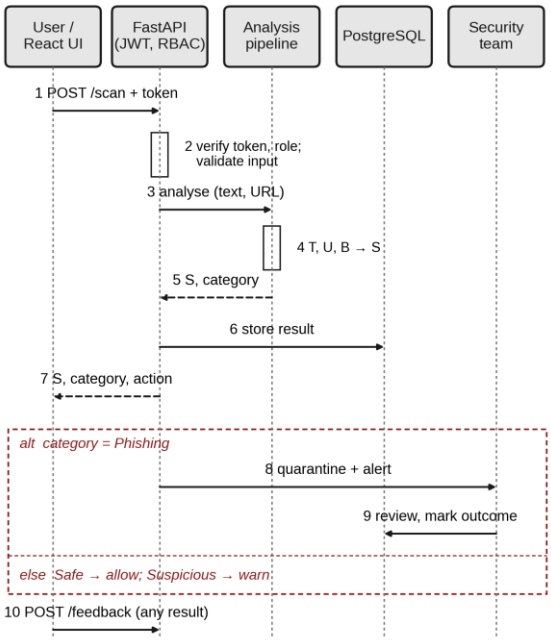


Fig. 4. Sequence of an authenticated scan, protective action, and feedback submission.

V. TESTING AND EVALUATION APPROACH

A. Functional testing

The project report outlines functional checks for input validation, data collection, feature extraction, classification, risk scoring, security actions, authentication, role-based access, dashboard reporting, database operations, and error handling. Positive tests should include valid URLs and messages; negative tests should include empty or malformed input, unusually long content, special characters, invalid credentials, expired tokens, and unauthorized requests.

B. Detection-quality protocol

A reproducible evaluation should use a labelled dataset separated into training, validation, and test partitions, with no leakage from near-duplicate URLs or messages. Hyper-parameters, weights, and thresholds are tuned on the validation partition only; the test partition is used once for final reporting. Table III lists the minimum metrics to report, per class.

Table III Minimum metrics for reproducible evaluation of PhishX.

Metric	Definition	Purpose
Precision	TP / (TP + FP)	Cost of false alarms
Recall	TP / (TP + FN)	Attacks actually caught
F1-score	2PR / (P + R)	Balance of both
False-positive rate	FP / (FP + TN)	Safe items wrongly flagged
False-negative rate	FN / (FN + TP)	Phishing items missed
Confusion matrix	Counts per class pair	Error structure

C. System and usability testing

API latency should be measured under documented load, and tests should verify whether warnings, quarantine, and alerts actually occur as expected. Usability testing should assess whether users understand the labels and recommended actions.

D. Reporting status

The source report describes test scenarios and intended behaviours but does not provide a complete dataset description or numerical test results. Therefore, no accuracy, response-time, or user-satisfaction figure is asserted in this manuscript. These measurements remain necessary before claims about effectiveness can be supported.

VI. SECURITY AND LIMITATIONS

A. Security and privacy

PhishX processes potentially sensitive message content, URLs, sender information, and account data. Data collection should be limited to what is needed, stored data should be protected, and access should be restricted by role. A valid token must not automatically grant permission to every record or administrative action. Logs should avoid retaining secrets or full message contents.

B. Limitations

Detection can fail when attackers adapt their language or infrastructure, when training data is unrepresentative, or when signals are missing or misleading. The sender-score method is not fully specified, the weights and thresholds are uncalibrated prototype values, and the report documents no benchmark. Feedback-driven retraining also requires review and data-quality controls to prevent incorrect or malicious reports from degrading the model. Human review should remain available for uncertain cases.

VII. CONCLUSION AND FUTURE WORK

This paper presented PhishX, an AI-assisted phishing-defense framework that combines message analysis, URL features, sender-related signals, risk scoring, classification, and protective actions. The described implementation uses React, Tailwind CSS, FastAPI, Scikit-learn, PostgreSQL, and JWT-based access control. Its central design goal is to connect detection with a response (allow, warn, or quarantine and alert) while retaining a feedback pathway for future improvement.

The current project documentation does not establish quantitative detection performance or production effectiveness. Future work should evaluate the system on current, labelled phishing and benign datasets; calibrate score weights and thresholds; measure false positives and false negatives; and test end-to-end response time, security, and usability. Planned extensions include browser integration, Gmail and

Outlook integration, QR-code phishing detection, attachment analysis, external threat-intelligence feeds, multilingual NLP, and behavioural analytics.

DECLARATION

A generative AI tool was used solely for grammar checking of the manuscript text. The authors reviewed all content and take full responsibility for the manuscript.

REFERENCES

[1]. Sahoo, D., Liu, C., and Hoi, S. C. H., "Malicious URL detection using machine learning: A survey," arXiv preprint arXiv:1701.07179, 2017.

[2]. Fette, I., Sadeh, N., and Tomasic, A., "Learning to detect phishing emails," in Proceedings of the 16th International Conference on World Wide Web (WWW), 2007.

[3]. Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S., "A comparison of machine learning techniques for phishing detection," in Proceedings of the APWG eCrime Researchers Summit, 2007.

[4]. Sahingoz, O. K., Buber, E., Demir, O., and Diri, B., "Machine learning-based phishing detection from URLs," Expert Systems with Applications, vol. 117, pp. 345–357, 2019.

[5]. Jain, A. K., and Gupta, B. B., "Towards detection of phishing websites on client-side using machine learning-based approach," Telecommunication Systems, vol. 68, 2018.

[6]. Salton, G., and Buckley, C., "Term-weighting approaches in automatic text retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.

[7]. Pedregosa, F., et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[8]. Jones, M., Bradley, J., and Sakimura, N., "JSON Web Token (JWT)," RFC 7519, Internet Engineering Task Force, 2015.

[9]. Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E., "Role-based access control models," IEEE Computer, vol. 29, no. 2, pp. 38–47, 1996.

[10]. FastAPI, "FastAPI documentation."

[11]. React, "React documentation."

[12]. Tailwind CSS, "Documentation."

[13]. PostgreSQL Global Development Group, "PostgreSQL documentation."

[14]. Mubaraali, L., Kuppuswamy, N., and Muthukumar, R., "Internet of Things-based induction motor diagnosis using convolutional neural network," Electric Power Components and Systems, vol. 52, no. 2, pp. 265–276, 2024, doi: 10.1080/15325008.2023.2217434.

[15]. Mohan Raj, C., Rajan Babu, W., Uday, K., Senthilkumar, R., Sudha, V., and Anandhakumar, V., "Smart dimensional measurement and material transportation (SDMMT) system using artificial intelligence," in Proceedings of the International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2023, pp. 493–497, doi: 10.1109/IDCIoT56793.2023.10053429.

[16]. Davis, J., and Goadrich, M., "The relationship between precision-recall and ROC curves," in Proceedings of the 23rd International Conference on Machine Learning (ICML), 2006, pp. 233–240.