

AI Cyber-Shield an Explainable Machine-Learning Framework for Phishing and Quishing Detection

Rahul Jadhav¹, Mahesh Rane²

¹Student, Sonopant Dandekar Arts, V.S. Apte Commerce and M.H. Mehta Science College

²Assistant Professor, Sonopant Dandekar Arts, V.S. Apte Commerce and M.H. Mehta Science College

Abstract—Phishing remains a major cybersecurity threat in which attackers use fraudulent websites, emails, and messages to steal passwords, banking information, and other sensitive data. A newer variant, Quishing or QR phishing, hides malicious destinations inside QR codes and makes manual inspection difficult for ordinary users. This paper presents AI Cyber-Shield, a web-based phishing and Quishing detection framework that unifies URL analysis, QR-code decoding, and email analysis in a single system. The proposed system combines rule-based security indicators with a Python-based machine-learning layer supporting Random Forest, XGBoost, Support Vector Machine (SVM), and ensemble classification. URL features include HTTPS usage, domain structure, URL length, suspicious keywords, excessive symbols, IP-based URLs, subdomains, shorteners, redirects, brand-related spelling errors, and possible typo squatting. QR images are decoded without opening or executing their embedded links, while email analysis examines sender information, subject, message body, embedded links, urgent language, account-verification requests, impersonation indicators, and other social-engineering patterns. The framework produces a SAFE, SUSPICIOUS, or PHISHING classification, a risk score from 0 to 100, detected warning indicators, an explanation, and a recommended action. Machine-learning performance is intended to be evaluated using Accuracy, Precision, Recall, F1-Score, and a Confusion Matrix. The design emphasizes unified detection and understandable results rather than relying on a single input type or an unexplained binary prediction.

Index Terms—Phishing Detection, Quishing, QR Phishing, Cybersecurity, Machine Learning, Random Forest, XGBoost, Support Vector Machine, Ensemble Classification, Typo squatting, Explainable Security.

I. INTRODUCTION

Phishing is a common cybersecurity attack in which criminals create deceptive websites, emails, and messages to obtain passwords, banking details, and personal information. Attackers increasingly use shortened URLs, brand impersonation, typo squatting, unusual domain structures, suspicious IP-based URLs, and social-engineering language to make malicious content appear legitimate. Quishing extends the same threat model to QR codes. Because a QR image conceals its destination until it is decoded, users may be redirected to a fraudulent or harmful website without being able to inspect the URL beforehand. AI Cyber-Shield is designed as a unified web-based cybersecurity system for analyzing URLs, QR codes, and emails. Instead of giving only a classification, the system also generates a risk score and understandable warning indicators. The current design combines rule-based security analysis with a Python machine-learning implementation that can use Random Forest, XGBoost, SVM, and an ensemble classifier. The principal goal is to provide a practical browser-accessible tool that identifies suspicious inputs while explaining the security indicators that influenced the result.

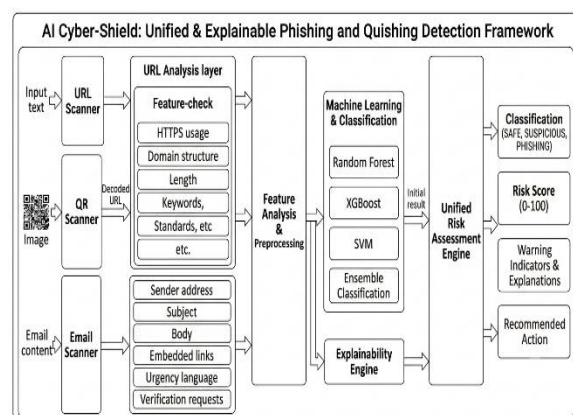
II. LITERATURE REVIEW

Existing phishing-detection approaches commonly analyze URLs, email content, or known malicious indicators. URL-oriented methods can inspect lexical and structural properties such as URL length, HTTPS usage, suspicious tokens, excessive symbols, IP addresses, subdomains, shortening services, and domain characteristics. Machine-learning approaches

extend this idea by learning patterns from labeled phishing and legitimate URL datasets. Algorithms such as Random Forest, SVM, and gradient-boosting methods can classify feature vectors extracted from URLs. Email phishing analysis focuses on sender information, message subjects, body text, embedded links, urgency, verification requests, impersonation, and other social-engineering signals. Quishing introduces an additional challenge because the suspicious content is encoded visually in a QR image. A practical Quishing detector therefore needs to decode the QR content safely before applying URL analysis, without automatically opening the destination. The project references Phish Tank, Open Phish, Kaggle phishing URL datasets, Scikit-learn, XGBoost, OpenCV, QR-code security research, typo squatting research, social-engineering research, and OWASP guidance as relevant technical and data sources. The research gap addressed by AI Cyber-Shield is the need for one explainable platform that combines URL, QR-code, and email analysis rather than treating these inputs as separate user workflows

III. SYSTEM ARCHITECTURE

The proposed architecture accepts three user input types: a URL, a QR-code image, or email information. The URL Scanner passes the entered URL to a feature-analysis layer that checks HTTPS usage, domain structure, length, suspicious keywords, excessive symbols, IP addresses, subdomains, URL shorteners, redirects, brand-related spelling errors, and possible typo squatting. The QR Scanner receives an image, detects and decodes the QR code, determines whether the decoded content contains a URL, and forwards that URL for security analysis. A key safety property is that the decoded link is analyzed without being opened or executed. The Email Scanner analyzes the sender address, subject, body, embedded links, urgent language, account-verification requests, impersonation indicators, and other social-engineering patterns. Results are retained for scan history and dashboard statistics. On edge devices.



IV. METHODOLOGY

A. URL Collection and Analysis - The system extracts security-relevant properties from a submitted URL. Analysis includes HTTPS usage, domain structure, URL length, suspicious keywords, excessive symbols, IP-based addressing, subdomains, URL shorteners, redirects, spelling variations associated with brands, and possible typo squatting. **B. QR-Code Processing** - The user uploads a QR-code image. The system detects and decodes the QR content using computer-vision/QR-processing functionality. If a URL is present, the decoded URL is passed to the URL-analysis pipeline. The link is not opened or executed during this process. **C. Email Analysis** - The system examines sender address, subject, message body, embedded URLs, urgency-related wording, account-verification requests, impersonation indicators, and other social-engineering patterns. **D. Result Generation** - For each scan, the system produces a security class, risk score, threat category, detected indicators, explanation, and safety recommendation. Scan information can be stored so that users can review previous results and dashboard statistics

V. IMPLEMENTATION

The proposed system is designed for Windows, Linux, or macOS and can be developed using Visual Studio Code. The browser interface uses React with TypeScript and Vite, together with HTML, CSS, and JavaScript. The machine-learning backend uses Python 3.11 or later with Flask. Scikit-learn and XGBoost provide the planned machine-learning algorithms, while OpenCV and QR-processing libraries support QR-code image analysis. SQLite is

specified for the Python backend, while the current React application uses browser Local Storage for scan history. The hardware requirements are intentionally modest: a computer or laptop with at least 4 GB RAM, with 8 GB recommended, an internet connection for development and testing, and a camera or QR-code image files for QR testing. This configuration supports the project's objective of providing a practical web-based cybersecurity dashboard rather than requiring specialized hardware.

TABLE I: Hardware Specifications and Training Hyperparameters.

Parameter / Component	Specification / Value
Primary GPU	CPU / Standard Workstation / Laptop (Min. 4 GB RAM, 8 GB Recommended)
Backend Framework	Python 3.11+, Flas
Frontend Framework	React with TypeScript, Vite, HTML, CSS, JavaScript
Machine Learning Libraries	Scikit-learn, XGBoost
Computer Vision / Processing	OpenCV, QR-code Processing Libraries
Supported ML Models	Random Forest, XGBoost, Support Vector Machine (SVM), Ensemble Classification
Data Sources	Phish Tank, Open Phish, Kaggle Phishing URL Datasets
Database Storage	SQLite (Backend), Browser Local Storage (Frontend History)
Performance / Output Targe	Multi-vector threat detection (URL, QR, Email) with 0–100 Risk Score & Explainability

VI. DISCUSSION

The principal strength of AI Cyber-Shield is its unified and explainable design. Users can analyze a conventional URL, a QR image that may contain a concealed destination, or suspicious email content from one system. The risk score and detected indicators are intended to help users understand why an input was considered dangerous rather than receiving an unexplained classification. The project also has important limitations. The current browser application primarily uses rule-based analysis and cannot guarantee detection of every new or zero-day phishing attack. Machine-learning accuracy depends on the quality, size, and diversity of the training data. QR detection depends on image quality, clarity, and format. The system does not open suspicious links, which improves safety but limits direct inspection of some website content and redirect behavior. Current browser scan history is local and is not automatically shared across devices. Protected pages, unavailable website content, or inaccessible redirects may reduce available evidence, while unusual but legitimate URLs or emails may occasionally be marked suspicious. External threat-intelligence services may also require internet connectivity, API keys, and additional configuration.

VII. CONCLUSION AND FUTURE SCOPE

AI Cyber-Shield presents a unified framework for phishing and Quishing detection across URLs, QR codes, and email content. The architecture combines explainable rule-based indicators with a machine-learning backend supporting Random Forest, XGBoost, SVM, and ensemble classification. Its three-level SAFE, SUSPICIOUS, and PHISHING output, 0–100 risk score, warning indicators, explanations, and recommendations are designed to make phishing detection more understandable to users. Future development can strengthen the system by training and validating the machine-learning models on larger and more diverse phishing datasets, improving cross-device scan-history storage, and integrating appropriately configured threat-intelligence services. Further work can also improve handling of website redirects, protected content, QR images of varying quality, and newly 2 emerging phishing patterns. Final experimental results should be

reported only after model testing using Accuracy, Precision, Recall, F1-Score, and Confusion Matrix

REFERENCES

- [1]. Tayachi, A., Ouni, B., Mourad, A., & Erbad, A. (2025). Quishing attack detection and mitigation using machine learning and deep learning for malicious URL identification. 2025 International Wireless Communications and Mobile Computing (IWCMC), Abu Dhabi, United Arab Emirates, pp. 1–6. <https://doi.org/10.1109/IWCMC65282.2025.11059621> Cited by: 4
- [2]. Nayak, G. S., Muniyal, B., & Belavagi, M. C. (2025). Enhancing phishing detection: A machine learning approach with feature selection and deep learning models. IEEE Access, 13, 33308–33320. <https://doi.org/10.1109/ACCESS.2025.3543738> Cited by: 84
- [3]. Remya, S., Pillai, M. J., Nair, K. K., Rama Subbareddy, S., & Cho, Y. Y. (2024). An effective detection approach for phishing URL using ResMLP. IEEE Access, 12, 79367–79382. <https://doi.org/10.1109/ACCESS.2024.3409049> Cited by: 57
- [4]. Zara, U., Ayyub, K., Ullah Khan, H., Daud, A., Alsahfi, T., & Gulzar Ahmad, S. (2024). Phishing website detection using deep learning models. IEEE Access, 12, 167072–167087. <https://doi.org/10.1109/ACCESS.2024.3486462> Cited by: 67