

PhishCatcher: A Machine Learning-Based Client-Side Defence Against Web Spoofing Attacks

Pragati R. Pujar, B. Sona, M K. Chithra, S. Darshan, N N. Deepthi, Bhagyashree Wakde

Department of Computer Science and Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, Karnataka, India

Published October 2026

Abstract

Phishing and web spoofing attacks continue to be a major cybersecurity challenge, deceiving users into revealing confidential information through fraudulent websites that mimic trusted domains. Conventional defences such as server-side monitoring, heuristic filters, and blacklist-based detection often fail to identify newly emerging or zero-day phishing domains, leaving end-users exposed. To address this gap, PhishCatcher proposes a client-side defence framework that integrates machine learning classifiers for real-time detection of spoofed websites. The system extracts a diverse set of discriminative features, including lexical analysis of URLs, structural attributes of HTML content, SSL certificate metadata, and visual similarity measures between suspected and legitimate sites. These features are processed locally on the user's device, enabling proactive classification without reliance on external threat intelligence feeds. Experimental evaluation demonstrates that PhishCatcher achieves high detection accuracy with low false-positive rates, while maintaining lightweight computational overhead suitable for browser integration. By combining resilience against zero-day phishing domains with efficient client-side deployment, PhishCatcher enhances user trust in online interactions and contributes to a safer digital ecosystem.

Keywords: phishing detection; web spoofing; client-side defence; machine learning; zero-day attacks; browser integration; cybersecurity resilience; endpoint protection

1 Introduction

The rapid growth of online services has transformed the way individuals interact with financial institutions, e-commerce platforms, and communication systems. However, this reliance on web applications has also created opportunities for adversaries to exploit user trust through phishing and web spoofing attacks. In such attacks, malicious actors design fraudulent websites that closely resemble legitimate ones, tricking unsuspecting users into disclosing sensitive information such as login credentials, banking details, or personal data. According to recent cybersecurity reports, phishing remains one of the most prevalent and damaging forms of cybercrime, with millions of new phishing domains registered annually.

Traditional defence mechanisms, including server-side monitoring and blacklist-based detection, provide partial protection but suffer from inherent limitations. Blacklists are reactive in nature and often fail to detect newly created or zero-day phishing domains. Server-side solutions, while effective in centralized environments, may not offer timely protection for end-users browsing the web independently. These shortcomings highlight the need for a more proactive and user-centric approach to phishing defence.

To address this challenge, we propose PhishCatcher, a client-side defence framework that leverages machine learning models to detect spoofed websites in real time. Unlike traditional methods,

PhishCatcher operates locally on the user's device, extracting discriminative features such as URL lexical patterns, HTML structural attributes, SSL certificate metadata, and visual similarity measures. These features are processed by trained classifiers to distinguish legitimate sites from malicious ones, thereby reducing reliance on external threat intelligence feeds.

Beyond the technical limitations of existing defences, phishing attacks exploit psychological factors such as urgency, fear, and trust, making them particularly effective against non-technical users. Attackers continuously adapt their strategies by registering new domains, employing homograph attacks, and leveraging HTTPS certificates to appear legitimate. This dynamic threat landscape underscores the importance of adaptive security mechanisms that can evolve alongside attacker techniques. By embedding detection capabilities directly on the client side, PhishCatcher not only provides immediate protection but also empowers users with transparent, real-time feedback about the legitimacy of the websites they visit.

Phishing detection has evolved through several generations of approaches. Early systems relied on blacklist-based detection, where known malicious domains were stored and blocked. While simple to implement, these methods are reactive and ineffective against newly registered or zero-day phishing domains. To improve responsiveness, heuristic approaches analyzed URL characteristics such as length, suspicious tokens, and domain age. Although lightweight, these methods often produced high false-positive rates, reducing user trust in the system.

Server-side monitoring frameworks introduced centralized intelligence, leveraging large datasets and threat intelligence feeds to identify phishing campaigns. These systems achieved strong detection accuracy but introduced latency and dependency on external servers, limiting their ability to provide immediate protection. Moreover, attackers frequently adapted their strategies to bypass server-side filters, reducing long-term effectiveness.

Another line of work explores visual similarity detection, comparing the rendered appearance of suspected websites with legitimate ones. While effective in identifying spoofed interfaces, these methods are computationally intensive and often impractical for real-time client-side deployment. Hybrid approaches combining lexical, structural, and visual features have shown promise, but integration into lightweight, user-centric tools remains limited. By integrating machine learning classifiers into a lightweight framework, PhishCatcher addresses the limitations of prior work and offers a proactive solution for end-users.

2 Literature survey

Phishing detection has been widely studied, with researchers proposing diverse strategies to mitigate web spoofing attacks. Early approaches relied on blacklist-based systems, where repositories of known malicious domains were maintained and updated. While effective against previously identified threats, these systems are reactive and fail to detect zero-day phishing domains. To improve responsiveness, heuristic methods analyzed URL characteristics such as unusual length, suspicious tokens, or domain age. Although lightweight, these methods often produced high false positives, limiting their practical adoption.

Subsequent research introduced server-side monitoring frameworks, leveraging centralized intelligence and large datasets to identify phishing campaigns. These systems achieved strong detection accuracy but introduced latency and dependency on external servers, reducing their ability to

provide immediate protection to end-users. Moreover, attackers frequently adapted their strategies to bypass server-side filters, diminishing long-term effectiveness.

Recent studies have focused on machine learning-based detection, where classifiers such as decision trees, support vector machines, and deep neural networks are trained on features extracted from URLs, HTML content, and SSL metadata. These approaches demonstrate improved adaptability and accuracy compared to traditional methods. However, most implementations remain server-centric, requiring continuous updates from centralized repositories. This dependency creates vulnerabilities when feeds are outdated or unavailable.

Another line of research explores visual similarity detection, comparing the rendered appearance of suspected websites with legitimate ones. While effective in identifying spoofed interfaces, these methods are computationally intensive and often impractical for real-time client-side deployment. Hybrid approaches combining lexical, structural, and visual features have shown promise, but integration into lightweight, user-centric tools remains limited.

Table 1. Literature survey on phishing detection approaches.

Approach	Era	Features	Strengths	Limitations	Studies	Relevance
Blacklist detection	2000s	Domain/IP lists	Simple, common	Misses zero-day phishing	Safe Browsing	Pushes proactive ML
Heuristic methods	2000s	URL length, tokens	Fast, lightweight	High false positives	CANTINA	Shows accuracy need
Server-side monitoring	2010s	Threat feeds, logs	Accurate, scalable	Latency, dependency	Cloud systems	Gap in client defence
ML classifiers	2010s–20s	URL, HTML, SSL	Adaptive, accurate	Server-centric, updates	SVM, Neural Nets	Basis for PhishCatcher
Visual similarity	2015+	Page rendering	Detects spoofed sites	Heavy computation	DOM/image hashing	Inspires hybrid features

The survey summarised in Table 1 reveals that while significant progress has been made, existing solutions often lack client-side defence mechanisms capable of providing immediate, proactive protection without reliance on external feeds. This gap motivates the development of PhishCatcher, which integrates machine learning classifiers into a lightweight client-side framework, offering resilience against evolving phishing strategies and empowering users with real-time security.

Phishing detection has progressed from early blacklist methods and heuristic approaches, which were simple and lightweight but prone to false positives and ineffective against zero-day phishing, to more advanced server-side monitoring and machine learning classifiers that improved accuracy but relied heavily on centralized updates. Later, visual similarity techniques and hybrid models combined multiple features for balanced detection, yet most remain resource-intensive or server-centric. This highlights the need for efficient client-side solutions, motivating the proposed PhishCatcher framework.

3 Proposed methodology

The proposed PhishCatcher architecture consists of five modules: user interaction, feature extraction, ML classification, rule-based analysis, and user alerts. When a user accesses a webpage, the system

extracts URL, HTML, SSL, and visual features, processes them using a lightweight TFLite model, and applies heuristic rules for validation. The results are securely transmitted via MQTT/TLS to Firebase or local storage, and alerts are displayed instantly through the Flutter interface. This design ensures client-side detection without heavy server dependency, while maintaining real-time responsiveness and security.

3.1 Overall system architecture

The overall PhishCatcher system architecture is designed as a client-side phishing detection framework that combines feature extraction, machine learning classification, and rule-based analysis to deliver real-time protection. The workflow begins with user interaction, where URLs and webpages are accessed. These inputs undergo feature extraction (URL, HTML, SSL, and visual attributes), which are then processed by a lightweight TFLite model for phishing detection. A rule-based layer validates suspicious patterns, ensuring robustness. The results are securely transmitted via MQTT/TLS to Firebase or local storage, while the Flutter interface provides immediate alerts and recommendations to the user. This architecture emphasizes client-side efficiency, security, and responsiveness, reducing reliance on server-side monitoring. The overall architecture is shown in Figure 1.

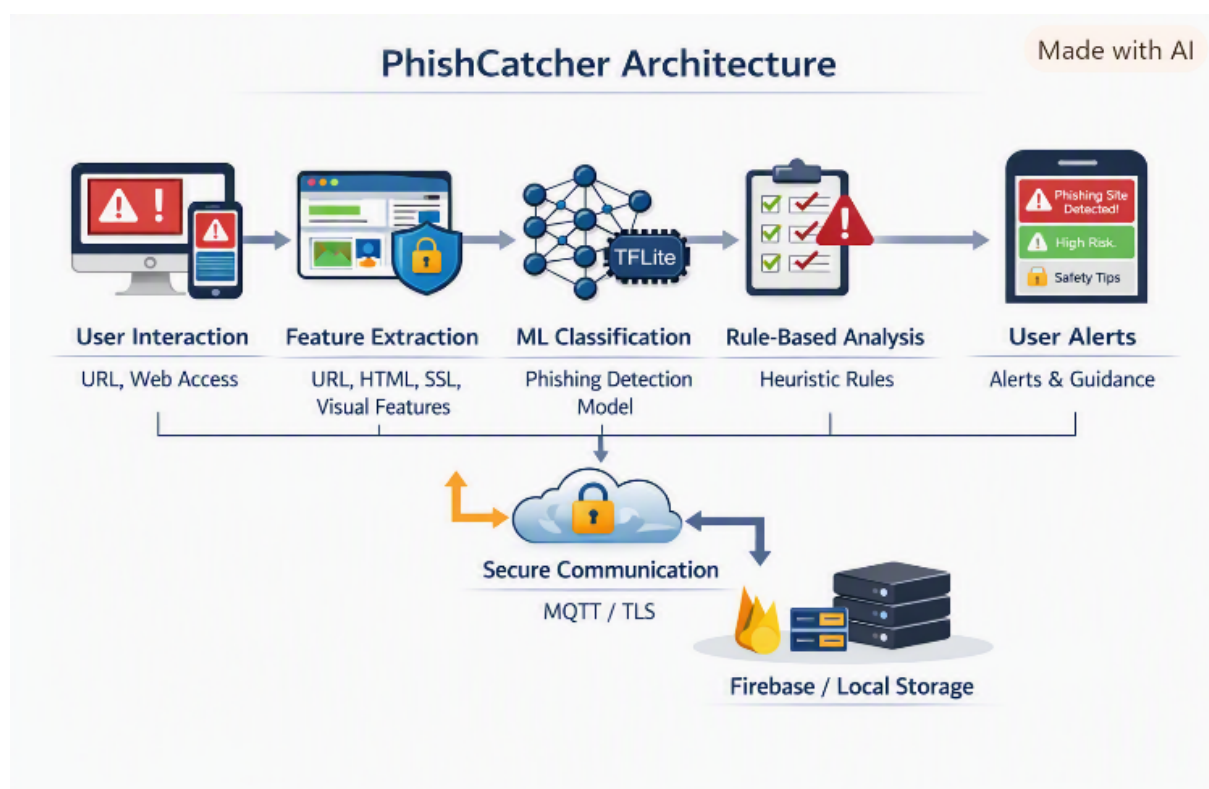


Figure 1. Overall system architecture of PhishCatcher.

4 Main components

The main components of the proposed system are described below.

4.1 Browser/client layer

- Runs as a browser extension or client-side application.

-
- Intercepts the URL before or during page loading.
 - Examines page characteristics without necessarily sending sensitive browsing data to a remote server.

4.2 Feature extraction layer

The system extracts several types of features:

4.2.1 URL features

- URL length
- Number of subdomains
- Presence of IP address instead of domain
- Suspicious characters
- HTTPS usage
- Domain age/reputation, if available
- Use of URL shortening
- Typosquatting similarity

4.2.2 HTML/DOM features

- Number and type of forms
- Password/input fields
- External links
- Hidden elements
- JavaScript behavior
- Login/payment forms
- Suspicious redirects

4.2.3 Visual features

- Page layout
- Logos and images
- Color/layout similarity
- Screenshot similarity to legitimate sites
- Visual characteristics of login pages

4.3 Feature preprocessing

- Removes irrelevant information.
- Converts extracted characteristics into numerical features.
- Normalizes the features before they are passed to the ML model.

4.4 Machine-learning detection engine

Possible algorithms include Random Forest, XGBoost, SVM, neural networks, CNNs, or lightweight transformer models, depending on the required accuracy and client hardware. The machine-learning detection engine is illustrated in Figure 2.

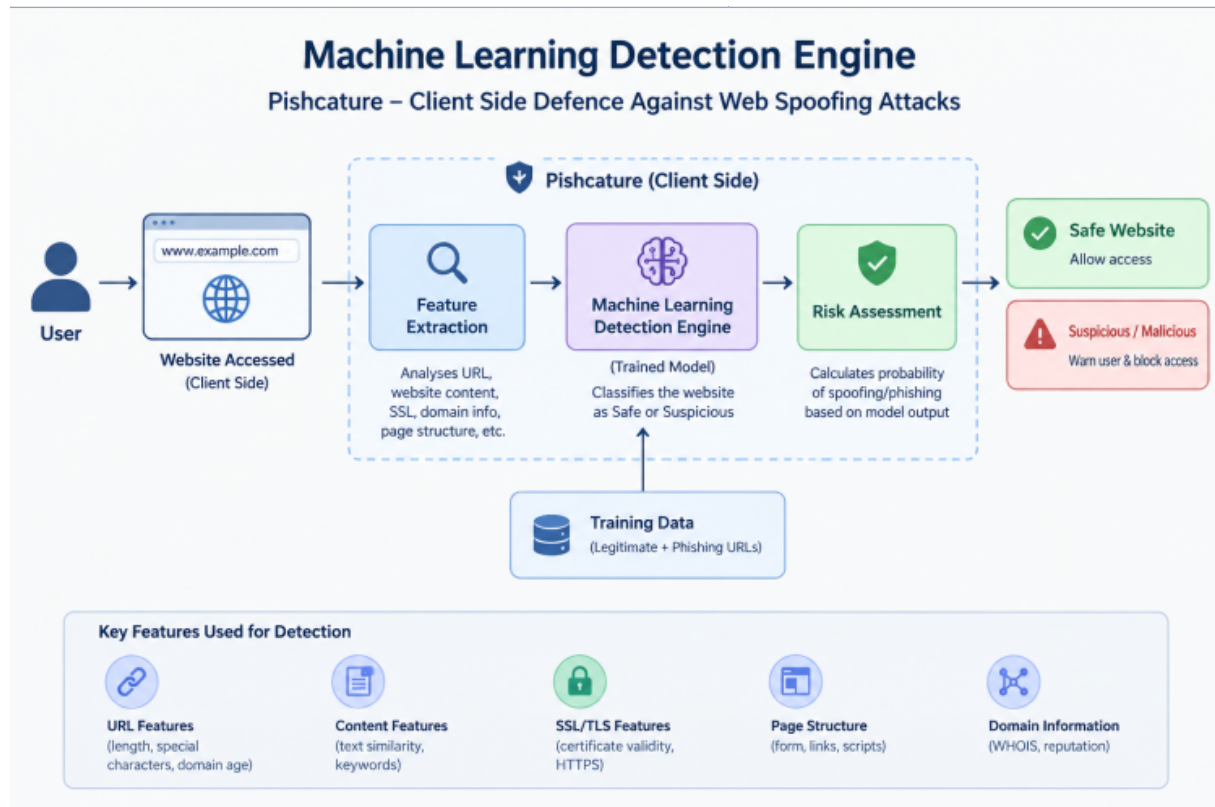


Figure 2. Machine-learning detection engine of the client-side system.

4.5 Risk/decision engine

The individual model outputs can be combined into a final risk score:

$$\text{Final risk score} = w_1 \cdot S_{\text{URL}} + w_2 \cdot S_{\text{DOM}} + w_3 \cdot S_{\text{Visual}} + w_4 \cdot S_{\text{Reputation}} \quad (1)$$

where w_1 – w_4 are the weights assigned to the URL, DOM, visual and reputation scores, respectively. The result can be classified as:

- 0–30%: Safe → allow
- 30–70%: Suspicious → display warning
- 70–100%: High risk → block the page

The exact thresholds should be determined experimentally using validation data rather than assumed beforehand.

4.6 Client-side defence layer

When a page is identified as malicious, PhishCatcher can:

- Prevent navigation.
- Display a warning page.
- Highlight suspicious login/payment fields.
- Warn the user before submitting credentials.
- Block suspicious redirects.
- Provide an option to report a false positive.

4.7 Model update layer

A practical architecture can periodically receive new model parameters or signatures from a trusted server while keeping real-time detection on the client.

The architecture can therefore be summarised as five major layers: browser → feature extraction → ML detection → risk analysis → defence.

4.8 System architecture

The architecture of PhishCatcher consists of several connected components.

4.8.1 Browser layer

This is where the user interacts with websites. The PhishCatcher extension operates alongside the browser and monitors web-page navigation.

4.8.2 Feature extraction layer

This component obtains relevant characteristics from the website.

The features can be divided into:

- URL features
- HTML/DOM features
- optional visual features
- optional domain/reputation features

4.8.3 Preprocessing layer

Raw features are cleaned and transformed into a format that can be processed by the ML model.

4.8.4 Machine learning layer

The trained classifier analyzes the feature vector and estimates the likelihood that the website belongs to the malicious class.

4.8.5 Risk assessment layer

The prediction is converted into an understandable security decision.

4.8.6 Defence layer

Depending on the risk level, PhishCatcher can:

- allow the webpage;
- display a warning; or
- prevent access to the webpage.

5 Expected outcome

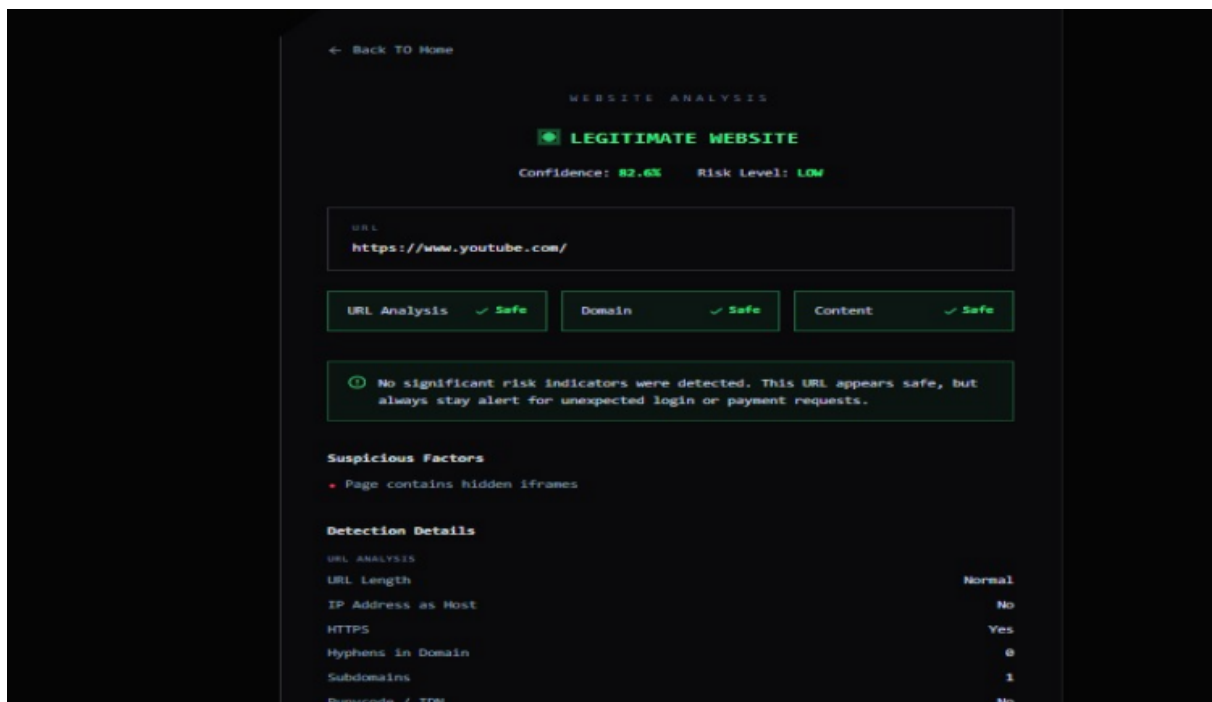


Figure 3. PhishCatcher analysis of a legitimate website.

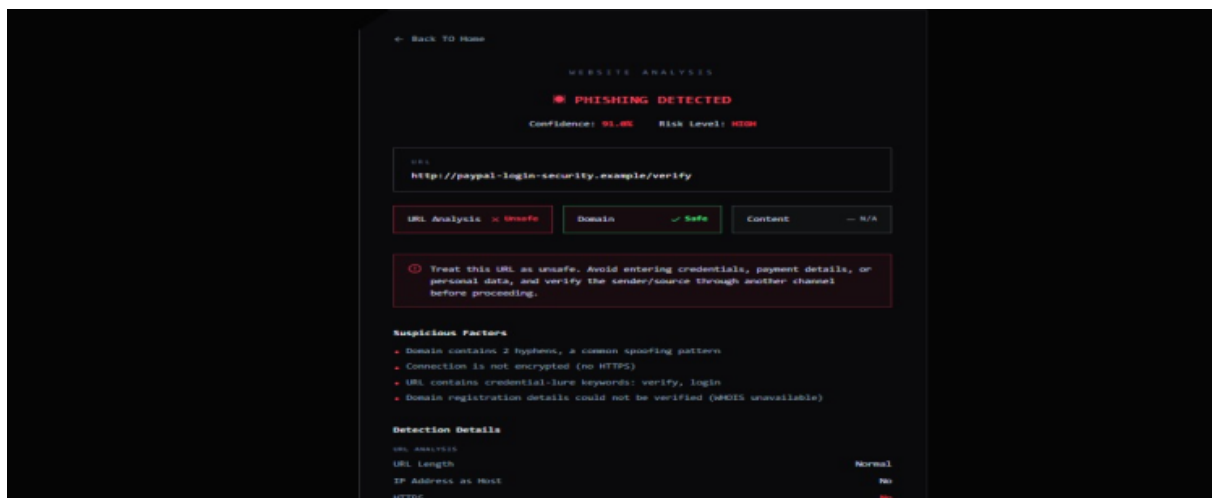


Figure 4. PhishCatcher detection of a phishing URL.

Figure 3 shows a legitimate website analysis performed by PhishCatcher. The system identifies the website as “LEGITIMATE WEBSITE” with a confidence of 82.6% and a low risk level. It analyzes the URL, domain, and website content and marks them as safe. The detection details show that the URL length is normal, it does not use an IP address as the host, it uses HTTPS, has no unusual hyphens, and does not use Punycode/IDN. Although a hidden iframe is detected as a suspicious factor, the overall analysis considers the website safe.

Figure 4 shows a phishing website detection. PhishCatcher identifies the URL as “PHISHING DETECTED” with a confidence of 91.0% and a high risk level. The system detects several suspicious factors, including multiple hyphens in the domain, the absence of HTTPS, and keywords such as “login” and “verify”, which can be associated with phishing attempts. The system therefore marks the URL

analysis as unsafe and warns the user not to enter credentials, payment details, or personal information. These two screenshots demonstrate how PhishCatcher uses multiple website characteristics and Machine Learning to distinguish between legitimate and potentially phishing websites.

These two screenshots demonstrate the working of our PhishCatcher system. In the first case, the system analyzes a legitimate website and classifies it as legitimate with 82.6% confidence and a low risk level. It checks different URL, domain, and content characteristics. In the second case, the system analyzes a suspicious URL and detects it as phishing with 91% confidence and a high risk level. It identifies factors such as multiple hyphens, lack of HTTPS, suspicious login and verification keywords, and unavailable domain registration information.

Therefore, instead of depending only on a blacklist, PhishCatcher examines multiple characteristics and uses Machine Learning to estimate the risk of a website."The expected outcome of the project is a functional browser-based prototype capable of identifying potentially spoofed websites using machine learning.

The completed system should demonstrate:

- extraction of relevant website features;
- machine-learning-based classification;
- real-time or near-real-time risk assessment;
- browser-based warnings;
- blocking of high-risk websites; and
- measurable detection performance.

The project should also provide experimental evidence showing how well the selected model distinguishes legitimate websites from phishing or spoofed websites.

6 Conclusion

This project presented PhishCatcher, a client-side system designed to provide additional protection against web spoofing and phishing attacks using machine learning.

The project addresses the difficulty users face when distinguishing genuine websites from fraudulent websites. Instead of depending entirely on known malicious URL lists, PhishCatcher examines characteristics of the website and uses a trained machine-learning model to estimate its risk.

The project also demonstrates that machine-learning performance should not be judged using accuracy alone. Precision, recall, F1-score, false-positive rate, detection time, and resource consumption are all relevant when developing a practical security system.

Overall, PhishCatcher demonstrates the potential of combining machine learning with browser-based security to provide users with an additional layer of protection against web spoofing.

References

- [1] Omotosho A, Balogun H, Yi Y. Phishing URL detection and interpretability with machine learning: A cross-dataset approach. *Security and Privacy*. 2025. <https://doi.org/10.1002/spy2.70175>
- [2] Kavya S, Sumathi D. Staying ahead of phishers: A review of recent advances and emerging methodologies in phishing detection. *Artificial Intelligence Review*. 2025;58:50. <https://doi.org/10.1007/s10462-024-11055-z>

-
- [3] Mousavi M, et al. Phishing website detection: An in-depth investigation of feature selection and deep learning. *Expert Systems*. 2025. <https://doi.org/10.1111/exsy.13824>
 - [4] Yoon JH, Buu SJ, Kim HJ. Phishing webpage detection via multi-modal integration of HTML DOM graphs and URL features based on graph convolutional and transformer networks. *Electronics*. 2024;13(16):3344. <https://doi.org/10.3390/electronics13163344>
 - [5] Phishing URL detection using machine learning and deep learning. In: Proceedings of the 2024 IEEE World AI IoT Congress (AIoT). 2024. <https://doi.org/10.1109/AIoT61789.2024.10579005>
 - [6] An integrated model based on deep learning classifiers and pre-trained transformer for phishing URL detection. *Future Generation Computer Systems*. 2024;161:269–285. <https://doi.org/10.1016/j.future.2024.06.031>
 - [7] Sahingoz OK, Buber E, Demir O, Diri B. Machine learning based phishing detection from URLs. *Expert Systems with Applications*. 2019;117:345–357. <https://doi.org/10.1016/j.eswa.2018.09.029>
 - [8] Li Y, Yang Z, Chen X, Yuan H, Liu W. A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems*. 2019;94:27–39. <https://doi.org/10.1016/j.future.2018.11.004>
 - [9] Hasane Ahammad SK, Kale SD, Upadhye GD, Pande SD, Babu EV, Dhumane AV, Jang Bahadur DK. Phishing URL detection using machine learning methods. *Advances in Engineering Software*. 2022;173:103288. <https://doi.org/10.1016/j.advengsoft.2022.103288>
 - [10] Aljofey A, et al. An effective detection approach for phishing websites using URL and HTML features. *Scientific Reports*. 2022;12:8842. <https://doi.org/10.1038/s41598-022-10841-5>
 - [11] Aljabri M, et al. URL phishing detection using machine learning techniques based on URLs lexical analysis. In: *Proceedings of the 12th International Conference on Information and Communication Systems (ICICS)*. 2021. <https://doi.org/10.1109/ICICS52457.2021.9464539>
 - [12] URL-based phishing websites detection via machine learning. In: Proceedings of the International Conference on Data Analytics for Business and Industry (ICDABI). 2021. <https://doi.org/10.1109/ICDABI53623.2021.9655851>
 - [13] Phishing websites detection using machine learning with URL analysis. In: Proceedings of the IEEE World Conference on Applied Intelligence and Computing (AIC). 2022. <https://doi.org/10.1109/AIC55036.2022.9848895>
 - [14] A comparative analysis of machine learning-based website phishing detection using URL information. In: *Proceedings of the 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*. 2022. <https://doi.org/10.1109/PRAI55851.2022.9904055>
 - [15] Phishing detection system through hybrid machine learning based on URL. *IEEE Access*. 2023;11:36805–36822. <https://doi.org/10.1109/ACCESS.2023.3252366>