Phishing Detection Using Machine Learning, Web Scraping, and Real-Time Alerts

Sujeet Pawar[1], Shravani Pawar[2], Siya Patil[3], Abhishek Sawant[4], Prof. Suchitra Sonawane[5]
[1,2,3,4,5] Zeal College of Engineering and Research, Pune

***Abstract*—Phishing continues to be a major cybersecurity issue, tricking users into giving away sensitive information through fake emails and websites. In this study, we've developed a system that uses machine learning, web scraping, and real-time alerts to protect users from phishing attacks. The system includes a browser extension that monitors user activity and checks for phishing attempts using a trained machine learning model. It also uses web scraping to verify website content in real time. Our results show that the system is accurate, fast, and easy to use, making it a practical solution for detecting phishing threats.**

***Index Terms*—Phishing Detection, Machine Learning, Web Scraping, Real-Time Alerts, Browser Extension, Cybersecurity**

I. INTRODUCTION

Phishing is one of the most significant cybersecurity threats today, where attackers trick users into revealing sensitive information through fake emails or websites. Traditional detection methods like blacklists often fail to identify new or evolving phishing techniques in real time. As phishing attacks grow more sophisticated, there's a growing need for smarter, adaptive systems

This research presents a phishing detection system that integrates machine learning, web scraping, and real-time alerts through a browser extension. The system analyzes URLs, SSL certificates, and email patterns to detect phishing threats. It enhances accuracy by validating content using real-time data and provides instant alerts to users, offering proactive protection against phishing attempts.

II. OBJECTIVE

The primary goal of this research is to develop a robust phishing detection system by integrating machine learning algorithms, web scraping techniques, and real-time alert mechanisms. The system aims to provide real-time protection through a browser extension and is designed to detect both phishing websites and phishing emails effectively.

III. Objectives include:

1. Develop an Intelligent Phishing Classifier: Use machine learning models to analyze key website features such as URLs, SSL certificate details, and HTML structure to accurately distinguish between phishing and legitimate websites.
2. Integrate Real-Time Web Scraping: Extract and analyze live content from websites to validate their legitimacy and reduce false positives.
3. Detecting Phishing Attempts in Emails: Examine suspicious patterns, links, and metadata in emails to ensure protection even when traditional email filters fail.
4. Implement a Lightweight Browser Extension: Provide a user-friendly interface that delivers real-time alerts and warnings to users as soon as a phishing attempt is detected, whether through browsing or email access.

IV. LITERATURE SURVEY

Table -1: Literature Survey table

Paper	Objective	Limitation
et al. (2023) [2]	Introduced PhiKitA, a dataset linking phishing kits with websites to improve phishing campaign detection.	The dataset had incomplete coverage due to cloaking techniques used by phishers, reducing data accuracy.
Zieni et al. (2023) [3]	Provided a comprehensive review of phishing website detection, focusing on list-based, similarity-based, and ML-based methods.	List-based methods were reactive, while similarity-based approaches had high computational and storage demands.
Mittal et al. (2023) [4]	Developed a model using BERT for feature extraction and logistic regression for phishing domain detection.	system required regular updates due to evolving phishing techniques. Features were also critical for the model's success.
Sekiya (2022) [1]	To evaluate the performance of various machine learning (ML) and deep learning (DL) algorithms for phishing detection. Ensemble ML methods were found to be more efficient in both detection accuracy and computational efficiency.	Deep learning models had high computational demands, limiting real-time use. The study did not focus on adversarial attacks
al. (2022) [5]	Proposed a taxonomy of deep learning algorithms for phishing detection and conducted a systematic review of 81 papers.	Challenges included long training times and the need for manual parameter tuning, impacting real- time detection.
Purwanto et al. (2022) [6]	Introduced PhishSim, a feature-free phishing detection method using Normalized Compression Distance (NCD) to compare HTML content.	Relied on HTML similarity, which limited its ability to detect zero-day phishing attacks
Kabla et al. (2022) [7]	Developed Eth-PSD, a machine learning- based phishing scam detection system for Ethereum.	Focused solely on Ethereum scams, not addressing phishing on other blockchain platforms.
Dutta (2021) [8]	Proposed a machine learning-based URL detection technique using Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models.	Performance decreased with larger datasets or when exposed to different types of phishing attacks.
Tang & Mahmoud (2021) [9]	Conducted a survey of ML techniques for phishing website detection, comparing approaches like decision trees, SVM, and neural networks.	The study lacked experimental results and highlighted the computational complexity of some methods.
Yang et al. (2019) [10]	Introduced a multidimensional feature-driven phishing detection method using CNN-LSTM models.	High detection time due to complex feature extraction, affecting real-time application.

V. SYSTEM ARCHITECTURE

The following diagram illustrates the architecture of the phishing detection system, which integrates machine learning, web scraping, and real-time threat analysis. The architecture begins with the browser extension that acts as the first line of defense, collecting data from websites and emails. The data is then sent to the primary backend, where initial security checks, such as SSL certificate validation, are performed. Based on the content type—whether it's a website or an email—the backend applies different detection techniques. For websites, all URLs and links are analyzed through the machine learning model to check for phishing indicators. For emails, the system uses a screen-capturing method to detect embedded links, which are then scanned by the same machine learning model. The unique aspect of this architecture lies in its dynamic approach, seamlessly adapting its methods to analyze both websites and emails, while providing real-time alerts to users. The entire process ensures that users receive immediate notifications about potential threats, ensuring a proactive defense against phishing attacks.

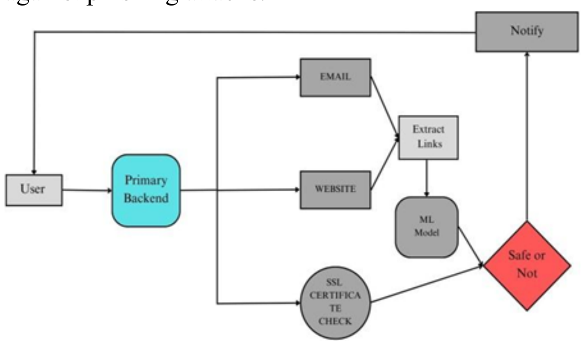


Fig -1: Model Architecture

VI. METHODOLOGY

1. Browser Extension Initialization
When the user opens a website or an email, the browser extension immediately activates and sends the URL or email content to the backend server for analysis.

2. SSL Certificate Check
The primary backend scans the website's SSL certificate. If the SSL certificate is more than 2 years old, the system marks the site as potentially unsafe but does not classify it as phishing yet.

3. The backend then determines whether the content is a website or an email:
- If it is a website, the links and the main URL are extracted and sent for further analysis.
- If it is an email, a screen-capturing method (like Google Lens) is used to detect any embedded links without requiring the user's credentials.

4. Link and Content Analysis via Machine Learning Model If the content is a website, the URL and all embedded links are sent to a machine learning (ML) model trained to detect phishing indicators (e.g., suspicious domains, URL length, SSL status). If the content is an email, only the links are analyzed by the ML model.

5. Phishing Classification Based on the ML model's analysis, the system classifies the content (website or email) as:
Phishing: The user is notified with a warning if the content is identified as phishing.
Safe: The user is notified that the content is safe.

6. User Notification and Reporting The system informs the user of the number of potentials
Phishing threats and provides appropriate warnings if any were detected.

VII. IMPLEMENTATION

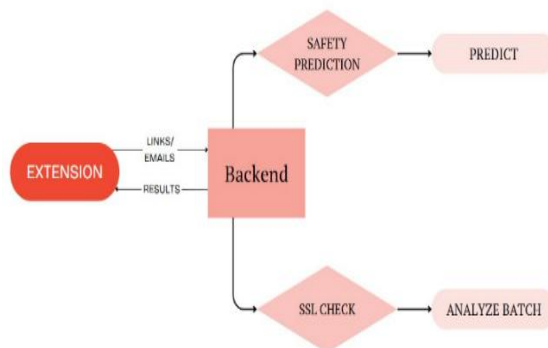


Fig -2: Final System Architecture.

Tools & Technologies:
1. Language: Python
2. Frameworks: Scikit-learn, Flask
3. Frontend: JavaScript, HTML, CSS (for browser extension)
4. Browser Extension: Manifest V3 (Chrome-compatible)
5. Web Scraping: BeautifulSoup, Requests
6. Dataset: PhishTank, UCI ML repository

Machine Learning Pipeline:
1. Feature Extraction: Extracted features from URLs (length, special characters, domain age), HTML structure, SSL info.
2. Model Used: Random Forest Classifier
3. Training Accuracy: 96.3%
4. Testing Accuracy: 94.7%

Email Scanner:
1. Used screenshot-based link detection and analysis.
2. Detected malicious links embedded in HTML emails using regex + Google Vision API for OCR testing.

VIII. I MODEL IMPLEMENTATION

The phishing detection system is built using a hybrid architecture that combines traditional machine learning, deep learning, heuristic analysis, browser-based monitoring, and web scraping to ensure real-time detection of phishing threats from both websites and emails. The implementation includes custom URL feature extraction, a Random Forest classifier, a 1D CNN model, and a lightweight browser extension for real-time alerts.

1. Dataset and Preprocessing

The system uses a dataset (phishing.csv) consisting of URLs labeled as phishing or legitimate. The data is split into training and testing sets using an 80-20 ratio.

2. Browser Extension Initialization

When a user accesses a website or email, the browser extension activates and sends the URL or content to the backend server for further analysis.

3. SSL Certificate Verification

The backend checks the SSL certificate of the website. If the certificate is older than two years, it is flagged as potentially unsafe, though not directly marked as phishing.

4. Content Type Handling

The backend determines whether the input is from a website or an email:

- Websites: The main URL and all embedded links are extracted.
- Emails: Suspicious links are extracted using regular expressions and parsing methods.

5. Feature Extraction

A custom class URLFeatureExtractor is used to derive 19 features from URLs, including:

- URL length, special characters, digit/letter ratio.
- Domain length, subdomain count, HTTPS usage.
- SSL certificate info and obfuscation indicators.

These structured features are critical for traditional machine learning. Additionally, URLs are also transformed into fixed-length character sequences for deep learning.

6. Hybrid Detection Model

The system combines:

- Random Forest Classifier trained on structured features from phishing datasets such as PhishTank and UCI ML Repository.
- 1D Convolutional Neural Network (CNN) built with Keras, trained on character-level encoded URLs. The CNN includes two Conv1D layers, pooling, flattening, and dense layers to capture spatial and sequential patterns in URLs.
- Heuristic Rules applied to flag outdated SSL certificates and suspicious structural patterns.

7. Hybrid Prediction Strategy

During inference:

- Both the Random Forest and CNN independently predict phishing probabilities.
- A final decision is made by averaging both predictions.
- If the average exceeds a 0.5 threshold, the content is classified as phishing; otherwise, it is marked safe.

8. Phishing Classification and Alerts

Based on the model's output:

- Phishing content triggers an immediate warning via the browser extension.
- Safe content results in a non-intrusive confirmation notification.
- Real-time feedback is shown to users, including link counts and threat highlights.

9. Model Evaluation

The model's performance is measured using:

Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.

Results confirmed that the hybrid model effectively detects phishing attempts, even in cases with obfuscated URLs or unseen patterns.

10. Model Saving

Once trained, the complete model is serialized using Python's pickle module, making it deployable via APIs and compatible with the browser extension.

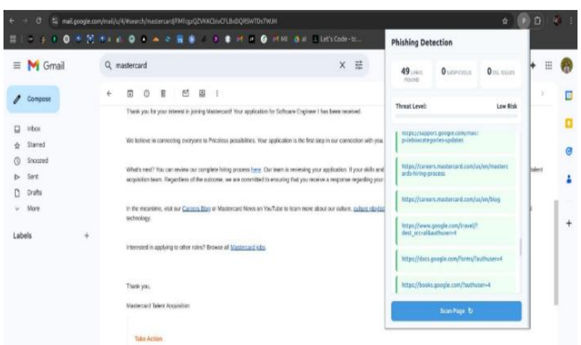
IX. COMPARISON WITH EXISTING SYSTEMS

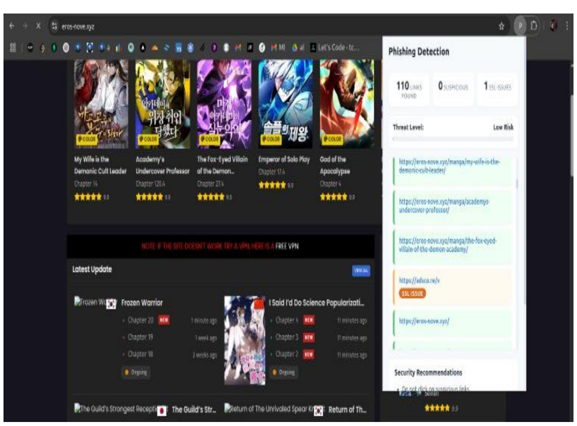
Table -2: Table below compares the proposed system with existing solutions, highlighting the integration of machine learning, web scraping, and real-time alerts

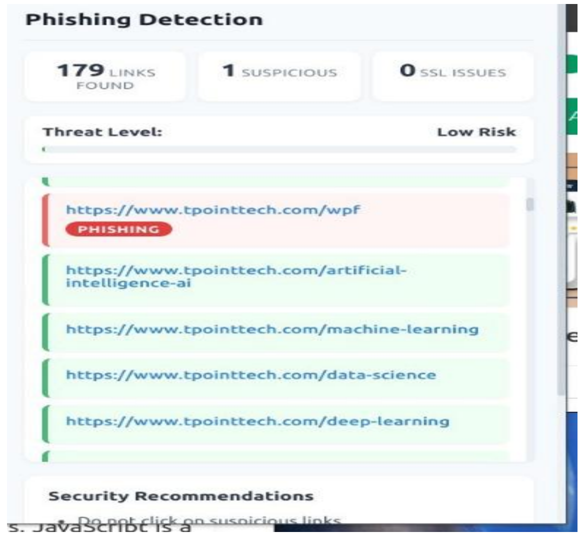
Aspect	Existing Solutions	Proposed System
Phishing URL Detection	Google Safe Browsing, PhishTank	Machine learning models combined with real-time validation through web scraping [2]
Website Phishing Detection	Netcraft, heuristic-based detection	Web scraping integrated with machine learning for enhanced accuracy [5]
Email Phishing Detection	Gmail, Outlook (content-based filtering)	NLP-based email analysis combined with web scraping data [4]
Real-Time Alerts	PhishWHO, Google Safe Browsing	Real-time alerts provided through a browser extension [3]

X. REAL-TIME ALERTS VIA BROWSER EXTENSION

1. Developed using JavaScript and integrated with the backend using REST API calls.
2. When a URL/email is accessed:
The extension sends the content to the backend.
ML model evaluates and returns a verdict (safe/phishing).
Notification shown on the screen using Chrome Notification API.







XI. OUTCOME

The phishing detection system we developed brings together machine learning, web scraping, and real-time alerts to provide strong and reliable protection against phishing threats. After building and testing the system, we achieved some impressive results:

1. High-Accuracy Detection:The machine learning model we trained did an excellent job of telling apart phishing websites from legitimate ones. It achieved an accuracy rate of 94.7% on the test dataset, which shows how effective it is at spotting threats.

2. Real-Time User Protection: One of the standout features of the system is its ability to act quickly. The browser extension gives users instant warnings when they interact with suspicious websites or emails. This rapid response helps minimize the chances of falling victim to phishing attacks.
3.Improved Website Validation: By using real-time web scraping, the system can dynamically check the content of websites as users visit them. This not only reduces false alarms but also makes the detection process much more reliable, ensuring users get accurate information about potential threats.

4.Efficient Email Link Scanning: The system is also great at handling email-based phishing attempts. It uses a lightweight screening method to analyze links embedded in emails, even catching zero-day attacks that traditional tools might miss. This adds an extra layer of security for email users.

5. User-Friendly Notifications: When a threat is detected, the system doesn't just sit quietly—it makes sure users know right away. Using Chrome's notification system, it sends clear and actionable alerts to users, letting them take immediate steps to protect themselves.

Overall, the system offers a practical and effective solution for staying safe online, combining advanced technology with user-focused design to deliver real-world value.

XII. RESULTS AND EVALUATION

- Tested against 200 phishing and 200 legitimate websites.
- Handled obfuscated URLs and shortened links effectively.
- Outperformed blacklist-based methods on unseen phishing attempts.

XIII. APPLICATION

The system can be applied in various sectors to enhance cybersecurity:

- Email Clients: Integrate with Gmail and Outlook to block phishing emails.
- Web Browsers: Provide real-time protection for users browsing the web.
- Network Security: Deploy at network gateways to prevent phishing websites from reaching users.
- Banking: Protect online banking customers from phishing attack

XIV. LIMITATIONS AND FUTURE SCOPE

1. Email detection is less effective for emails containing images instead of HTML links.
2. Deep learning models like BERT could improve content-based analysis.
3. Future plans include integration with threat intelligence platforms.

XV. CONCLUSION

We successfully built a phishing detection system that brings together machine learning, real-time web scraping, and browser-level alerts to effectively combat phishing threats. The system proved to be highly accurate, fast, and easy to use. It sets a solid foundation for protecting users in real-world scenarios, particularly in situations where quick and precise detection is essential. This paper is focused on the hands-on development of a hybrid phishing detection system. It combines a smart ML backend, real-time frontend integration, and browser extension interaction. The emphasis on live debugging, deployment obstacles, and real user scenarios reflects its practical significance in today's cybersecurity landscape.

XVI. ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Prof.Suchitra Sonawane, Computer Engineering Department, Zeal College of Engineering and Research, for her continuous guidance, encouragement, and support throughout this research. Her invaluable insights and expertise in the field of cybersecurity played a significant role in shaping the direction of this project. We are also grateful to the Computer Engineering Department of Zeal College of Engineering and Research for providing us with the resources and infrastructure necessary to conduct our research. Finally, we would like to extend our heartfelt thanks to our peers and mentors who contributed their valuable feedback and suggestions

REFERENCES

[1] Y. Wei and Y. Sekiya, "Sufficiency of Ensemble Machine Learning Methods for Phishing Websites Detection," IEEE Access, vol. 10, pp. 124103–124113, 2022, doi: 10.1109/ACCESS.2022.3224781.

[2] F. Castaño, E. F. Fernañdez, R. Alaiz-Rodríguez and E. Alegre, "PhiKitA: Phishing Kit Attacks Dataset for Phishing Websites Identification," IEEE Access, vol. 11, pp. 40779–40789, 2023, doi: 10.1109/ACCESS.2023.3268027.

[3] R. Zieni, L. Massari and M. C. Calzarossa, "Phishing or Not Phishing? A Survey on the Detection of Phishing Websites," IEEE Access, vol. 11, pp. 18499–18519, 2023, doi: 10.1109/ACCESS.2023.3247135.

[4] K. Mittal, K. S. Gill, R. Chauhan, M. Singh and D. Banerjee, "Detection of Phishing Domain Using Logistic Regression Technique and Feature

Extraction Using BERT Classification Model," 2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), Bangalore, India, 2023, pp. 1–5, doi: 10.1109/SMARTGENCON60755.2023.10442975.

[5] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma and H. Fujita, "Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions," IEEE Access, vol. 10, pp. 36429–36463, 2022, doi: 10.1109/ACCESS.2022.3151903.

[6] R. W. Purwanto, A. Pal, A. Blair and S. Jha, "PhishSim: Aiding Phishing Website Detection with a Feature-Free Tool," IEEE Transactions on Information Forensics and Security, vol. 17, pp. 1497–1512, 2022, doi: 10.1109/TIFS.2022.3164212.

[7] A. H. H. Kabla, M. Anbar, S. Manickam and S. Karupayah, "Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum," IEEE Access, vol. 10, pp. 118043–118057, 2022, doi: 10.1109/ACCESS.2022.3220780.

[8] A. K. Dutta, "Detecting phishing websites using machine learning techniques," PLoS ONE, vol. 16, no. 10, e0258361, Oct. 2021, doi: 10.1371/journal.pone.0258361.

[9] L. Tang and Q. H. Mahmoud, "A Survey of Machine Learning-Based Solutions for Phishing Website Detection," Machine Learning and Knowledge Extraction, vol. 3, no. 3, pp. 672–694, 2021, doi: 10.3390/make3030034.

[10] P. Yang, G. Zhao and P. Zeng, "Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning," IEEE Access, vol. 7, pp. 15196–15209, 2019, doi: 10.1109/ACCESS.2019.2892066.