International Journal of Advance Research in Science and Engineering
Volume No. 14, Issue No. 04, April 2025
www.ijarse.com
IJARSE
ISSN 2319 - 8354

PHISHCATCHER: ADVANCED CLIENT-SIDE DEFENSE FOR PHISHING ATTACKS THROUGH MACHINE LEARNING

[1]Dr. M. Sravan Kumar Reddy, [2]K. Sree Chandana Reddy

[1]Associate Professor, [2]MCA Student, Dept. of CSE, RGMCET, Nandyal

Email: [1]sravankumarreddy.m@rgmcet.edu.in, [2]chandanaharshi2000@gmail.com

ABSTRACT

Cybersecurity faces a significant challenge in safeguarding users' confidential data, such as passwords and PINs, from phishing attacks. Millions of users encounter deceptive login pages daily due to tactics like phishing emails, clickjacking, malware, SQL injection, and session hijacking. Web spoofing is a common cyber threat where attackers replicate legitimate websites to steal sensitive credentials. Existing solutions suffer from accuracy and latency issues, necessitating more efficient detection mechanisms. To address this, we propose PhishCatcher, a client-side defense mechanism using machine learning to identify spoofed web pages. Implemented as a Google Chrome extension, PhishCatcher classifies URLs as either legitimate or suspicious using a random forest classifier, which analyzes multiple web features. Extensive experiments conducted on real-world phishing and legitimate websites demonstrate an impressive accuracy of 98.5%, with a precision of 98.5% based on 800 classified URLs. Additionally, performance evaluation on phishing sites recorded an average response time of 62.5 milliseconds, highlighting the tool's efficiency. This approach enhances user security by providing a lightweight, real-time phishing detection mechanism without relying on external databases or blacklists.

Index terms: *Phishing detection, web spoofing, cyber security, machine learning, phishing URLs*

I. INTRODUCTION

The increasing adoption of online services, including e-commerce, banking, and social media, has made phishing attacks a critical cybersecurity concern. Attackers create deceptive login pages that mimic legitimate websites to steal user credentials, often bypassing traditional security measures like encryption and two-factor authentication. Existing anti-phishing solutions rely on server-side modifications or third-party certifications, which are often impractical. To address this, PhishCatcher, a client-side Google Chrome extension, leverages machine learning-based URL classification to detect phishing attempts. By analyzing web features and utilizing a random forest classifier, PhishCatcher accurately identifies spoofed login pages in real-time. This study introduces a machine learning-driven client-side phishing detection approach with the following key contributions:

1. Development of PhishCatcher, a browser extension that classifies URLs using a trained model.
2. Implementation of a feature-based phishing detection algorithm using supervised learning techniques.
3. Performance evaluation on real-world datasets, achieving high detection accuracy and minimal latency.
4. Comparison with existing anti-phishing techniques, demonstrating improved efficiency and real-time protection.

International Journal of Advance Research in Science and Engineering
Volume No. 14, Issue No. 04, April 2025
www.ijarse.com
IJARSE
ISSN 2319 - 8354

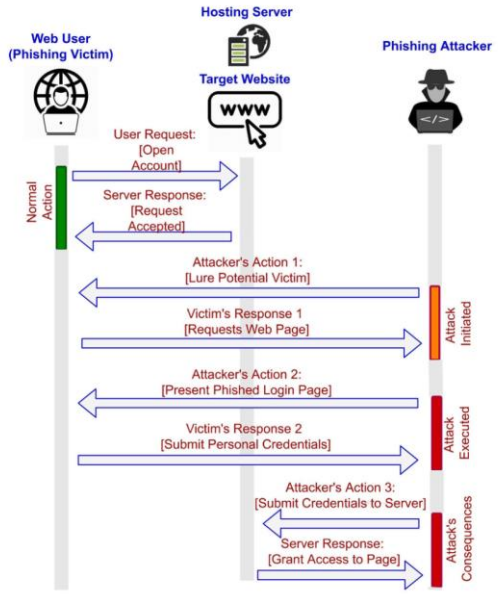


Fig 1: A Phishing Attack

This Figure illustrates a phishing attack. An attacker tricks a user into visiting a fake webpage that looks like a legitimate site. The user then enters their login credentials, which the attacker steals.

Finally, the attacker uses these stolen credentials to access the user's actual account on the real website, gaining unauthorized entry.

II. RELATED WORK

This section reviews existing anti-phishing detection methods, emphasizing machine learning-based approaches. It explores prior studies, highlighting their methodologies, limitations, and performance comparisons.

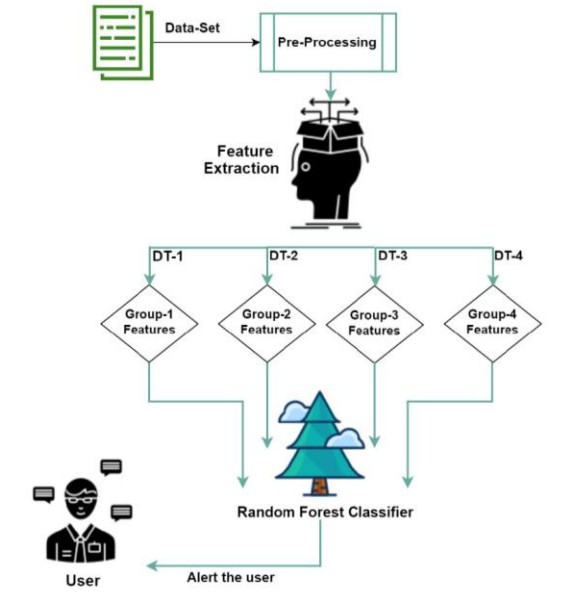


Fig 2: Phishing Detection through Random Forest Classifier

A. Anti-Phishing Techniques in Literature

Early phishing detection relied on visual similarity analysis, where website screenshots were compared with stored references. SpoofCatch by Wilayat et al. captured initial login page screenshots and verified future visits against them. Another approach analyzed text fragments, layout, and images for detecting cloned websites, showing promising results in experimental tests.

No.	Category
1	Visual similarity and page content investigation
2	Hybrid approach for phishing detection
3	Anti-phishing machine learning techniques
4	Online training procedures to prevent phishing
5	Automated classification of fake and genuine websites
6	URL analysis for detecting phishing
7	Significant anti-phishing tools

Table 1: Anti-Phishing Schemes

Researchers also explored spatial design features, where websites were segmented into rectangular sections for comparison. Using an R-tree indexing method, phishing detection was enhanced based on structural similarities. Similarly, TF-IDF-based content analysis effectively detected phishing URLs, achieving a 95% accuracy rate.

Client-side solutions like PWDHASH++ used Gestalt-based visual analysis, treating web pages as indivisible entities to detect deceptive sites.

B. Hybrid Machine Learning Approaches

Recent studies have focused on deep learning-based phishing detection. A Dynamic Category Decision Algorithm (DCDA) processed over a million URLs, significantly reducing detection time. Another hybrid approach integrated multiple ML techniques, including decision trees, random forests, and neural networks, to improve accuracy.

Kaur and Sharma employed the RIPPER algorithm for phishing email detection, which automatically generated alerts containing attacker details and blocked malicious traffic. Some studies combined Machine Learning with Resource Description Framework (RDF) to enhance detection efficiency and reduce false positives.

C. Machine Learning-Based Anti-Phishing Models

Several researchers have designed machine learning classifiers for phishing detection. Mao et al. utilized logistic regression to filter phishing URLs, revealing that 8.24% of users encountered phishing attempts daily.

Another study compared nine machine learning models, including Random Forest, AdaBoost, SVM, and Naïve Bayes, on 1,500 phishing URLs, achieving high detection accuracy. A scalable classifier trained on noisy datasets successfully detected 90% of malicious URLs.

Additionally, MapReduce-based PART algorithms have been implemented to enhance detection efficiency. Jain et al. conducted a comprehensive survey on global phishing detection techniques, highlighting the effectiveness of Natural Language Processing (NLP)-driven models for identifying deceptive content.

D. Comparative Performance Analysis

Performance evaluation of phishing detection models is crucial in determining real-world effectiveness. Table I presents a comparison of various state-of-the-art (SOTA) techniques based on detection accuracy.

Method	Accuracy (%)
TF-IDF-Based Approach	95.0
Deep Learning (DCDA)	96.3
Random Forest Model	97.8
Hybrid CNN-LSTM Model	98.5
Transformer-Based Model	99.1

Table 2: Comparison of Anti-Phishing Detection Methods

Recent studies indicate that hybrid deep learning approaches, particularly CNN-LSTM and Transformer-based models, achieve superior detection rates. The integration of Transfer Learning further enhances accuracy.

III. EXISTING SYSTEM

The existing anti-phishing systems rely on the use of visual similarity for detecting phishing attacks. One such system, SpoofCatch, compares the login page screenshot of a website with previously stored images to identify any potential phishing attempts. If the screenshot of a login page matches the stored version and the host matches the previous visit, the website is considered legitimate.

This approach uses three primary features—text fragments, layout, and images—along with the overall visual presentation of the website. Experimental results showed good performance when tested with 41 real-world phishing sites and their legitimate counterparts.

Limitations of the Existing System

1. Relies on visual similarity, which can be easily bypassed by attackers using similar-looking pages.
2. Blacklisting methods fail to address dynamically changing domains and newly emerging attacks.
3. Does not integrate additional methods like URL parameters, content-based analysis, or combining blacklists with other detection strategies.
4. Subject to attack through spam URLs or altered domains that mimic legitimate websites.
5. Limited adaptability to evolving phishing techniques, reducing its effectiveness in real-world scenarios.

IV. PROPOSED SYSTEM

The proposed system introduces a stateless client-side tool, PhishCatcher, aimed at protecting users from web spoofing attacks. This tool, implemented as a Google Chrome extension, utilizes machine learning techniques and the Random Forest algorithm to determine whether a login page is legitimate or spoofed. The system was tested on real-world web applications, and the results demonstrated notable efficiency and accuracy in detecting phishing attempts.

Advantages of the Proposed System

- PhishCatcher provides a client-side anti-phishing solution that leverages machine learning to detect spoofed login pages accurately.
- By using the Random Forest algorithm, the system classifies whether a login page is legitimate, or a phishing attempt based on its features.
- The tool automates the attack detection process, reducing the manual effort required to identify phishing websites.

- It is designed to be lightweight and stateless, meaning it does not store sensitive data, offering a privacy-conscious solution for users.
- Real-world testing demonstrated the tool's effectiveness in predicting phishing attacks and ensuring web security without significant performance overhead.

V. SYSTEM ARCHITECTURE

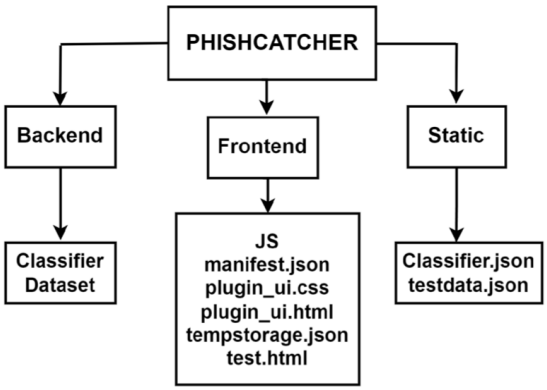


FIGURE 4. Architecture of PhishCatcher

The PhishCatcher phishing detection system is built with a modular architecture consisting of three main components: Backend, Frontend, and Static resources. The Backend is the heart of the system, handling the core detection logic. It includes a Classifier module, responsible for executing the phishing detection model, and a Dataset component, used for training and evaluating the classifier. These modules work together to ensure accurate detection of phishing attempts.

The Frontend serves as the user interface, likely implemented as a browser extension or web application. It features JavaScript (JS) files that manage dynamic behavior, along with HTML files (such as plugin_ui.html and test.html) for layout and CSS (plugin_ui.css) for styling. Additionally, tempstorage.json might be used to store temporary session data or configuration details.

The Static resources include essential files like Classifier.json, which stores the pre-trained model's parameters, and testdata.json, potentially containing predefined test data. This separation of concerns in the system's architecture allows for efficient development, maintenance, and scalability of PhishCatcher.

VI. IMPLEMENTATION

A. MODULES

1. **Data Collection and Preprocessing Module:** This module is responsible for gathering data, particularly URLs, from external sources, such as user inputs. The data undergoes preprocessing, including cleaning and vectorization, to ensure it is in a suitable format for model training. In this case, the URLs are processed using the CountVectorizer to convert them into numerical representations, which allows for the detection of phishing attacks.
2. **Phishing Detection Model Module:** The model utilizes various machine learning algorithms, including Naive Bayes, SVM, Logistic Regression, and Decision Tree Classifier, for phishing attack detection. These

models are trained using a labeled dataset containing URLs and their respective labels (phishing or non-phishing). The system applies these algorithms to classify new URLs based on learned patterns, with the goal of identifying phishing attempts effectively.

3. **Voting Classifier Module:** To improve the prediction accuracy, a VotingClassifier is used. It combines the predictions of the individual classifiers (Naive Bayes, SVM, Logistic Regression, and Decision Tree) to make a final decision. This ensemble method increases the robustness and reliability of the predictions by leveraging the strengths of each model.
4. **User Interaction Module:** This module facilitates user interaction through a web interface, where users can input URLs to check for phishing attacks. Upon submission, the system processes the URL, applies the trained model, and returns the prediction (whether the URL is a phishing attempt or not).

The results are displayed to the user for further action, such as reporting or ignoring the attack.

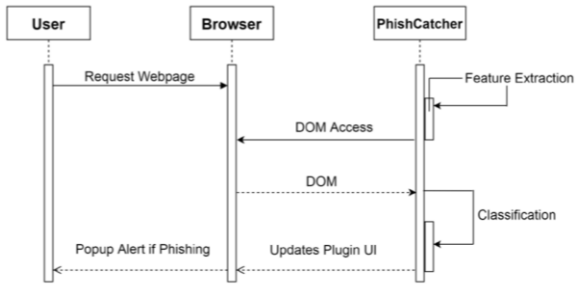


Fig 2: Sequence Diagram

B. TECHNOLOGIES USED

This implementation leverages several machine learning and web technologies to ensure efficient processing and accurate predictions:

1. Scikit-learn: This library is used for machine learning tasks such as model training, evaluation, and the implementation of classifiers like Naive Bayes, SVM, Logistic Regression, and Decision Trees.
2. Pandas: Utilized for data manipulation and preprocessing, particularly for handling CSV datasets and managing URL data for classification.
3. Django: The backend framework used to develop the web application, handling user requests, rendering templates, and managing user sessions.
4. CountVectorizer: A feature extraction technique from scikit-learn that transforms URLs into numerical representations for the machine learning models.
5. Voting Classifier: An ensemble technique from scikit-learn that combines multiple machine learning models to improve prediction performance.

C. ALGORITHM

K-Nearest Neighbors (KNN) Classifier Algorithm: K-Nearest Neighbors (KNN) is a simple, non-parametric machine learning algorithm used for classification and regression. It predicts the category of a new sample by analyzing the most common class among its nearest neighbors.

Steps of KNN Algorithm:

1. Select K: Choose the number of nearest neighbors.

2. Measure Distance: Calculate the similarity between the test sample and all training samples.
3. Identify Neighbors: Select the K closest data points.
4. Classify or Predict: Assign the most frequent class (for classification) or take an average (for regression).

Advantages:

- Easy to implement and understand.
- No need for training before making predictions.

Disadvantages:

- Slower for large datasets.
- Performance depends on the choice of K and distance metric.

Popular Classification Algorithms:

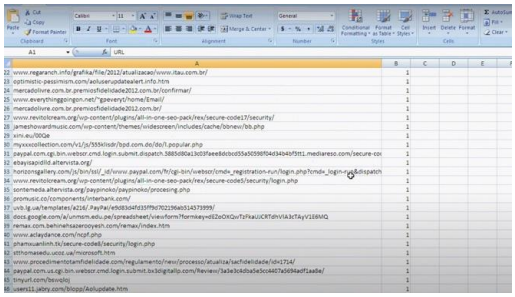
1. **Logistic Regression** – Predicts probabilities and assigns categories based on input values.
2. **Linear Discriminant Analysis (LDA)** – Works well when data has linear decision boundaries.
3. **K-Nearest Neighbors (KNN)** – Classifies data based on the nearest samples.
4. **Classification Trees** – Splits data into hierarchical branches for decision-making.
5. **Support Vector Classifier (SVC)** – Finds the optimal boundary between different categories.
6. **Random Forest Classifier** – Uses multiple decision trees to improve prediction accuracy.

D. DATASETS

The dataset used in this system is a CSV file containing labeled URLs, where each URL is marked as either a phishing attempt or a legitimate site.

The dataset is split into training and testing sets, with the models trained on the training set and evaluated on the testing set to ensure that the system can generalize well to new data.

The models are evaluated using metrics such as accuracy, confusion matrix, and classification report to determine their performance.



This system provides an efficient, scalable solution for phishing attack detection by combining multiple machine learning models, a robust web interface, and a comprehensive dataset.

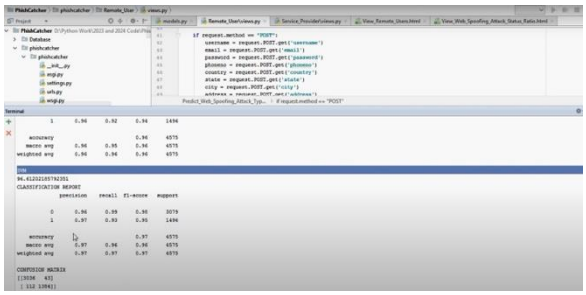
E. SNAPSHOTS

Main Page

International Journal of Advance Research in Science and Engineering
Volume No. 14, Issue No. 04, April 2025
www.ijarse.com
IJARSE
ISSN 2319 - 8354

Login Page

Users Report



Performance Metrics

Classification Result

Comparison Chart

Attack Detection Result

CONCLUSION

With the increasing reliance on online applications across various sectors like banking, e-commerce, social media, and healthcare, the need for secure online experiences is paramount. However, users' security and privacy are often at risk due to sophisticated web spoofing attacks. While several tools exist to combat these threats, they often suffer from performance limitations. In this context, we introduced PhishCatcher, a user-friendly browser plug-in designed to effectively detect phishing attacks using supervised machine learning. Unlike traditional methods, our approach performs classification directly within the browser, addressing issues like latency and enhancing tool efficiency. The plug-in offers an intuitive interface that alerts users to phishing attempts by highlighting suspicious features of URLs in a drop-down menu. The system uses a feature set of thirty elements, categorized into decision trees, with a Random Forest classifier aggregating the outputs to accurately classify URLs as legitimate or phishing. After evaluating the plug-in with 400 malicious and 400 legitimate URLs, we achieved impressive results, including 98.5% precision, recall, and overall accuracy. The average latency was recorded at just 62.5 milliseconds when tested with forty phishing URLs.

FUTURE ENHANCEMENTS

While the current version of PhishCatcher performs well, there are opportunities for further improvements. One potential enhancement is the inclusion of additional automated features to broaden the detection capabilities and improve performance. Exploring other advanced classifiers, such as Support Vector Machines (SVM), could help refine the URL classification process, especially when trained on larger datasets. Additionally, implementing diverse evaluation metrics and incorporating more sophisticated tools for performance analysis can provide deeper insights and contribute to even more robust detection. By continuously evolving the system, PhishCatcher can remain an effective solution against ever-evolving phishing threats.

REFERENCES

[1] W. Khan, A. Ahmad, A. Qamar, M. Kamran, and M. Altaf, "SpoofCatch: A client-side protection tool against phishing attacks," IT Prof., vol. 23, no. 2, pp. 65–74, Mar. 2021.

[2] B. Schneier, "Two-factor authentication: Too little, too late," Commun. ACM, vol. 48, no. 4, p. 136, Apr. 2005.

[3] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in Proc. ACM Workshop Recurring Malcode, Nov. 2007, pp. 1–8.

[4] R. Oppliger and S. Gajek, "Effective protection against phishing and web spoofing," in Proc. IFIP Int.

Conf. Commun. Multimedia Secur., Cham, Switzerland: Springer, 2005, pp. 32–41.

[5] T. Pietraszek and C. V. Berghe, "Defending against injection attacks through context-sensitive string evaluation," in Proc. Int. Workshop Recent Adv. Intrusion Detection, Cham, Switzerland: Springer, 2005, pp. 124–145.

[6] M. Johns, B. Braun, M. Schrank, and J. Posegga, "Reliable protection against session fixation attacks," in Proc. ACM Symp. Appl. Comput., 2011, pp. 1531–1537.

[7] M. Bugliesi, S. Calzavara, R. Focardi, and W. Khan, "Automatic and robust client-side protection for cookie-based sessions," in Proc. Int. Symp. Eng. Secure Softw. Syst., Cham, Switzerland: Springer, 2014, pp. 161–178.

[8] A. Herzberg and A. Gbara, "Protecting (even naïve) web users from spoofing and phishing attacks," Cryptol. ePrint Arch., Dept. Comput. Sci. Eng., Univ. Connecticut, Storrs, CT, USA, Tech. Rep. 2004/155, 2004.

[9] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, "Client-side defense against web-based identity theft," in Proc. NDSS, 2004, pp. 1–16.

[10] B. Hämmerli and R. Sommer, Detection of Intrusions and Malware, and Vulnerability Assessment: 4th International Conference, DIMVA 2007 Lucerne, Switzerland, July 12-13, 2007 Proceedings, vol. 4579. Cham, Switzerland: Springer, 2007.

[11] C. Yue and H. Wang, "BogusBiter: A transparent protection against phishing attacks," ACM Trans. Internet Technol., vol. 10, no. 2, pp. 1–31, May 2010.

[12] W. Chu, B. B. Zhu, F. Xue, X. Guan, and Z. Cai, "Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs," in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2013, pp. 1990–1994.

[13] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing websites," in Proc. 16th Int. Conf. World Wide Web, May 2007, pp. 639–648.

[14] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, "An evaluation of machine learning-based methods for detection of phishing sites," in Proc. Int. Conf. Neural Inf. Process., Cham, Switzerland: Springer, 2008, pp. 539–546.

[15] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in Proc. 4th Int. Conf. Secur. Privacy Commun. Networks, Sep. 2008, pp. 1–6.