Phishing Guard: Machine Learning- Powered Web Spoofing Defense

[1]Bhashaboina Shravya, [2]Dr.M.Dhanalakshmi
[1]*MCA Student, Department of Information Technology, Jawaharlal Nehru Technological University, India*
[2]*Professor of IT, Department of Information Technology, Jawaharlal Nehru Technological University, India*

Abstract: This project aims to tackle the ongoing threat of phishing attacks by developing Phishing Guard, a client-side defense tool. The main objective is to leverage machine learning as a foundational element for the effective identification of new web spoofing threats. By concentrating on the client side, the project intends to bolster the overall security against phishing attacks. The focus on machine learning highlights the necessity for an adaptable and intelligent defense system. By integrating machine learning into Phishing Guard, the project aims to equip the tool with the capability to outpace the constantly changing tactics used by phishing attackers. This strategy ensures a more efficient and responsive approach to newly emerging web spoofing threats.

Recognizing the growing risk posed by phishing, especially with the rise in online activities, this project emphasizes the critical need to combat web spoofing. The development of Phishing Guard is positioned as an essential measure to protect both user privacy and organizational security amidst increasing phishing threats. Unlike traditional server-side solutions that have inherent limitations, Phishing Guard adopts a client-side protection strategy. This choice allows users to gain from a comprehensive defense tool without requiring changes to the targeted websites. This client-side emphasis aims to overcome the limitations of conventional server-side solutions. Phishing Guard is designed with the end-user in mind, particularly those who are frequently targeted by phishing attacks. The tool provides significant advantages by enhancing online safety, drastically reducing the risk of identity theft, and preventing fraud through the effective detection of malicious URLs. By focusing on user protection, Phishing Guard becomes a valuable resource in strengthening individuals against the widespread threat of phishing attacks. We enhanced our anti-phishing tool by integrating Support Vector Machine, XGBoost, and a Stacking Classifier, thereby augmenting the system's capabilities. Additionally, a Flask framework with SQLite was implemented, providing streamlined signup and signin processes for user testing and input validation.

Index terms - Web spoofing, security and privacy, machine learning, web security, browser extension.

1. INTRODUCTION

With the rapid advancement in modern technology, online platforms like e-commerce, online banking, distance learning, e-health, and e-governance have significantly expanded. Social networking applications like Facebook and Twitter play a pivotal role in globalization, attracting billions of users. Many websites require users to create personalized accounts for a tailored experience, typically through login pages where users register with a username and password. However, this process poses significant privacy risks, particularly in the context of phishing attacks. A typical phishing scenario begins with a deceptive email containing a link to a malicious website, disguised to resemble a legitimate one. When the user unknowingly enters their credentials on this fake site, the attacker captures their information and uses it to access the legitimate site, leading to identity theft and compromised confidential information.

Identity theft, online fraud, and scams have surged with the rise of web spoofing and phishing attacks. These cybercrimes involve malicious actors stealing valuable data from users. Initially used for identity theft, web spoofing has evolved to target sensitive information related to national security, intellectual property, and organizational secrets. Modern phishing techniques include QR code phishing, mobile application spoofing, and spear phishing, which can bypass traditional security measures like firewalls,

digital certificates, encryption, and two-factor authentication.

To deceive victims, attackers often use logos and HTML from legitimate sites to create convincing fake websites. Common phishing vectors include email, trojans, keyloggers, and man-in-the-middle proxies, with online banking, third-party payment systems, and e-commerce sites being frequent targets. Since phishing attacks exploit non-cryptographic components, security protocols like SSL/TLS are insufficient on their own and require additional protective measures.

These protections can be implemented on the server-side or client-side. While server-side solutions can effectively identify spoofed sites, they are often neglected due to the effort required. Client-side solutions offer protection without server support and are the focus of this discussion. Anti-spoofing tools, which rely on third-party certification, passwords, or URLs, are classified as stateful or stateless and use either blacklists or heuristics for phishing detection.

Blacklist-based tools have high accuracy with minimal false positives, detecting around 90% of phishing sites, but they struggle with zero-day attacks and evolving threats. Heuristic-based tools, like CANTINA and SpoofCatch, can identify 90% of phishing sites with a 1% false positive rate. However, stateful techniques like SpoofCatch suffer from performance degradation over time as local storage fills with images of login pages, slowing down the comparison process and increasing detection latency.

We design a stateless anti-phishing tool using machine learning (ML). Over the past decade, researchers have used ML techniques to detect malicious URLs and prevent scams. These methods rely on training data to build models based on statistical properties, which predict whether a URL is malicious or safe. Key ML algorithms like Naïve Bayes, Support Vector Machines (SVM), and Logistic Regression (LR) are commonly used, but they have certain vulnerabilities that need to be addressed.

2. LITERATURE SURVEY

Most anti-phishing solutions either escape certain attack patterns or rely on complex parameters. We propose a simpler approach: preventing phishing attacks by focusing on the overall visual appearance of web pages. We introduce SpoofCatch, a browser extension that uses four similarity algorithms to compare the visual similarity between genuine and phished web pages. Extensive experiments show that SpoofCatch effectively captures all phishing attacks with minimal overhead[27].

Phishing is form of identity theft that combines social engineering techniques and sophisticated attack vectors to harvest financial information from unsuspecting consumers. Often a phisher tries to lure her victim into clicking a URL pointing to a rogue page. In this paper [3], we focus on studying the structure of URLs employed in various phishing attacks. We find that it is often possible to tell whether or not a URL belongs to a phishing attack without requiring any knowledge of the corresponding page data. We describe several features that can be used to distinguish a phishing URL [12, 34] from a benign one. These features are used to model a logistic regression filter that is efficient and has a high accuracy. We use this filter to perform thorough measurements on several million URLs and quantify the prevalence of phishing on the Internet today.

Phishing and Web spoofing have proliferated and become a major nuisance on the Internet. The attacks are difficult to protect against, mainly because they target non-cryptographic components, such as the user or the user-browser interface. This means that cryptographic security protocols, such as the SSL/TLS protocol, do not provide a complete solution to tackle the attacks and must be complemented by additional protection mechanisms. In this paper [4], we summarize, discuss, and evaluate the effectiveness of such mechanisms against (large-scale) phishing and Web spoofing attacks [4, 8].

Injection vulnerabilities, such as SQL injection, cross-site scripting, and shell injection, are significant threats to application security. Existing defenses depend heavily on developers, making them error-prone. In this paper, we introduce CSSE, a method to detect and prevent injection attacks by addressing the root cause: the ad-hoc serialization of user input. CSSE enforces separation of input channels using metadata and context-sensitive string evaluation, requiring no developer interaction or code modifications. We implemented a CSSE prototype for PHP and successfully validated it with the phpBB application, detecting and preventing all tested SQL injection attacks with minimal runtime overhead.

Additionally, "Session Fixation" vulnerabilities enable attackers to hijack sessions by controlling session identifiers, allowing them to impersonate victims. We analyze this issue and propose three server-side countermeasures tailored to real-life scenarios to mitigate these vulnerabilities.

3. METHODOLOGY

i) Proposed Work:

The proposed methodology for Phishing Guard involves utilizing machine learning, specifically the random forest algorithm. This algorithm dynamically assesses login web pages, discerning between legitimate and potentially spoofed sites. [28, 30, 32, 33] Phishing Guard employs a client-side defense mechanism, functioning as a Google Chrome extension. It leverages real-time learning, reducing reliance on historical data and enhancing adaptability to novel phishing tactics. The focus on client-side protection ensures compatibility without requiring modifications to targeted websites. In an effort to bolster the system's performance, we introduced additional machine learning classifiers, namely Support Vector Machine (SVM), XGBoost, and a Stacking Classifier, complementing the existing Random Forest algorithm. This ensemble approach, featuring a combination of [24, 35] RandomForest, ExtraTrees, and XGBoost classifiers in a stacking configuration, aims to enhance the robustness of our phishing detection system. Furthermore, a user-friendly Flask framework with SQLite integration has been developed, facilitating user testing through a seamless signup and signin process. This feature enables comprehensive evaluation and comparison of the classifiers' effectiveness, contributing to the overall refinement of the anti-phishing tool.

ii) System Architecture:

The system architecture begins with the selection of a suitable dataset, from which features are extracted based on their performance. These features are then input into multiple classifiers, including the Random Forest, SVM [14, 31], XGBoost, and Stacking Classifier extensions. These classifiers play a pivotal role in identifying and classifying phishing URLs. In the Phishing Guard browser extension, upon detecting a potential phishing attack, the system triggers an alert notification, actively enhancing users' overall security awareness.

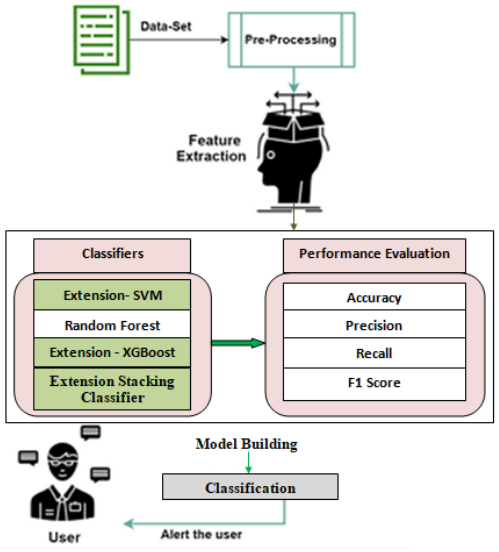

Fig 1 Proposed architecture

iii) Dataset collection:

To train propose algorithm we ahve used PHISHTANK [5, 55] dataset which contains 1000's of normal and phishing URL and by using this dataset we can predict URL as SAFE or phishing. Apart from training author has developed CHROME based extension which will analyse all visiting URLS and then alert user with SAFE or phishing URL'S.

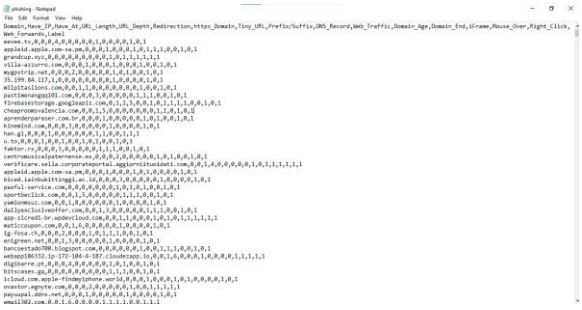

Fig 2 PHISHTANK dataset

iv) Data Processing:

Data processing involves transforming raw data into valuable information for businesses. Generally, data scientists process data, which includes collecting, organizing, cleaning, verifying, analyzing, and converting it into readable formats such as graphs or documents. Data processing can be done using three methods i.e., manual, mechanical, and electronic. The aim is to increase the value of information and facilitate decision-making. This enables businesses to

improve their operations and make timely strategic decisions. Automated data processing solutions, such as computer software programming, play a significant role in this. It can help turn large amounts of data, including big data, into meaningful insights for quality management and decision-making.

v) Feature selection:

Feature selection is the process of isolating the most consistent, non-redundant, and relevant features to use in model construction. Methodically reducing the size of datasets is important as the size and variety of datasets continue to grow. The main goal of feature selection is to improve the performance of a predictive model and reduce the computational cost of modeling. Feature selection, one of the main components of feature engineering, is the process of selecting the most important features to input in machine learning algorithms. Feature selection techniques are employed to reduce the number of input variables by eliminating redundant or irrelevant features and narrowing down the set of features to those most relevant to the machine learning model. The main benefits of performing feature selection in advance, rather than letting the machine learning model figure out which features are most important.

vi) Algorithms:

Support Vector Classifier (SVC): SVC, a robust classification algorithm, identifies a hyperplane in the feature space to maximize class separation. It plays a crucial role in this project by discerning patterns within the dataset, facilitating effective classification of URLs as phishing or legitimate based on extracted features [35].

```
# Support Vector Classifier model
from sklearn.svm import SVC
svc = SVC()

# fitting the model for grid search
svc.fit(x_train, y_train)
#predicting the target value from the model for the samples
y_train_svc = svc.predict(x_train)
y_test_svc = svc.predict(x_test)
```

Fig 3 SVC

Random Forest (RF): Random Forest, an ensemble learning algorithm, constructs multiple decision trees during training and outputs the mode of class predictions. In this project, RandomForestClassifier enhances the model's ability to manage intricate features and patterns, improving URL classification [24].

```
# Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(random_state=5)

# fit the model
forest.fit(x_train,y_train)

#predicting the target value from the model for the samples
y_train_forest = forest.predict(x_train)
y_test_forest = forest.predict(x_test)
```

Fig 4 Random forest

XGBoost (eXtreme Gradient Boosting): XGBoost, known for efficiency, builds sequential weak learners, usually decision trees, to create a potent predictive model. In this project, XGBClassifier serves as the final estimator, augmenting predictive power by effectively amalgamating outputs from various base models.

```
from xgboost import XGBClassifier

# instantiate the model
xgb = XGBClassifier()

# fit the model
xgb.fit(x_train,y_train)

#predicting the target value from the model for the samples
y_train_gbc = xgb.predict(x_train)
y_test_gbc = xgb.predict(x_test)
```

Fig 5 XGboost

Stacking Classifier: Stacking Classifier, an ensemble technique, combines base models using a meta-learner. It amalgamates predictions from RandomForestClassifier, Extra Trees classifier and LGBMClassifier in this project, leveraging their distinct attributes for superior accuracy in URL classification. This ensemble approach enhances predictive performance by blending strengths from various algorithms.

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.neural_network import MLPClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import StackingClassifier


RF = RandomForestClassifier(random_state=5)

estimators = [('rf', LGBMClassifier()),('et', RF)]

clf = StackingClassifier(estimators=estimators, final_estimator=XGBClassifier())


clf.fit(x_train,y_train)

#predicting the target value from the model for the samples
y_train_stac = clf.predict(x_train)
y_test_stac = clf.predict(x_test)
```

Fig 6 Stacking classifier

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones

classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

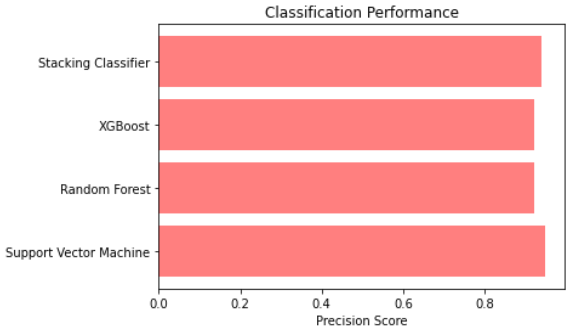

Fig 7 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN}$$

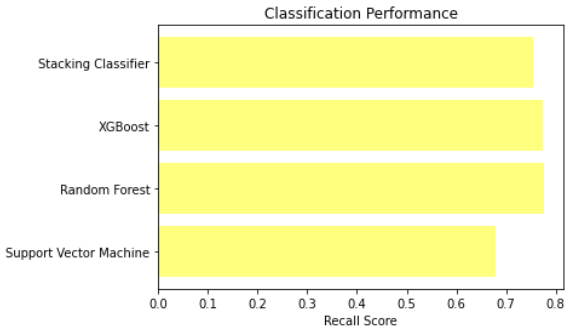

Fig 8 Recall comparison graph

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall correctness of a model's predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

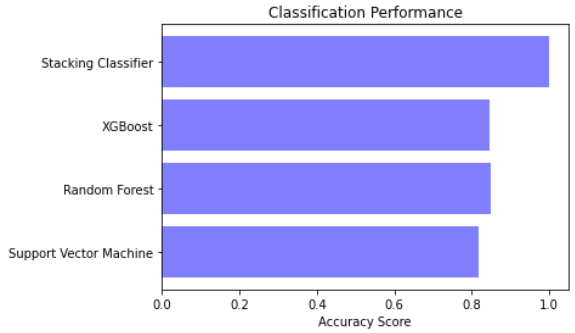

Fig 9 Accuracy graph

F1 Score: The F1 Score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives, making it suitable for imbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

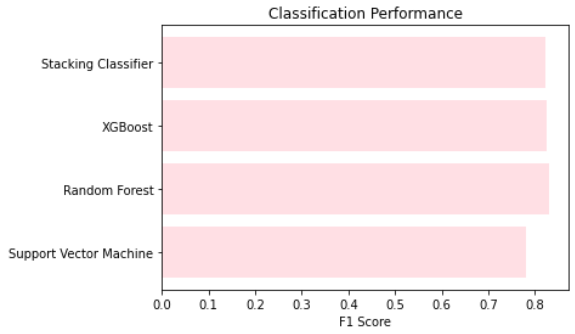

Fig 10 F1Score

	ML Model	Accuracy	f1_score	Recall	Precision	Specificity
0	Extension- SVM	0.817	0.780	0.680	0.949	0.975
1	Random Forest	0.850	0.831	0.777	0.923	0.953
2	Extension- XGBoost	0.846	0.826	0.775	0.921	0.951
3	Extension- Stacking Classifier	1.000	0.824	0.754	0.941	0.967

Fig 11 Performance Evaluation

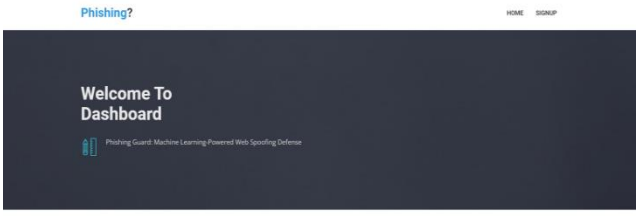

Fig 12 Home page

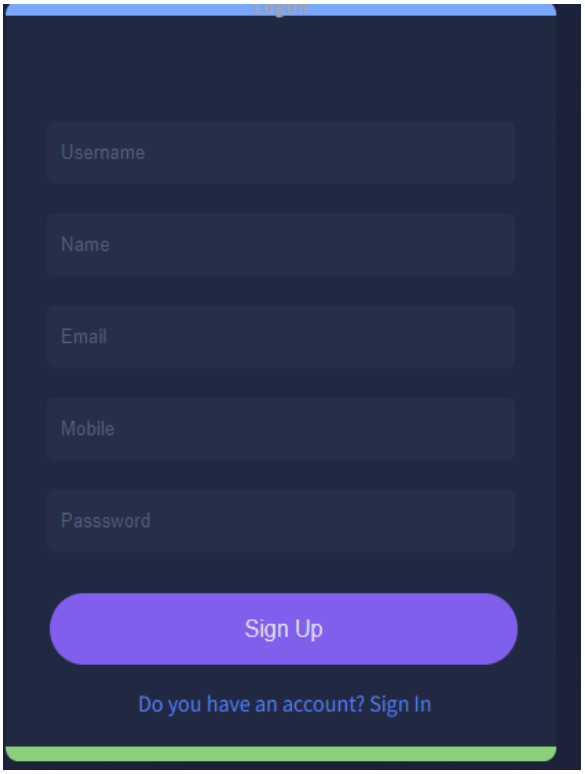

Fig 13 Signin page

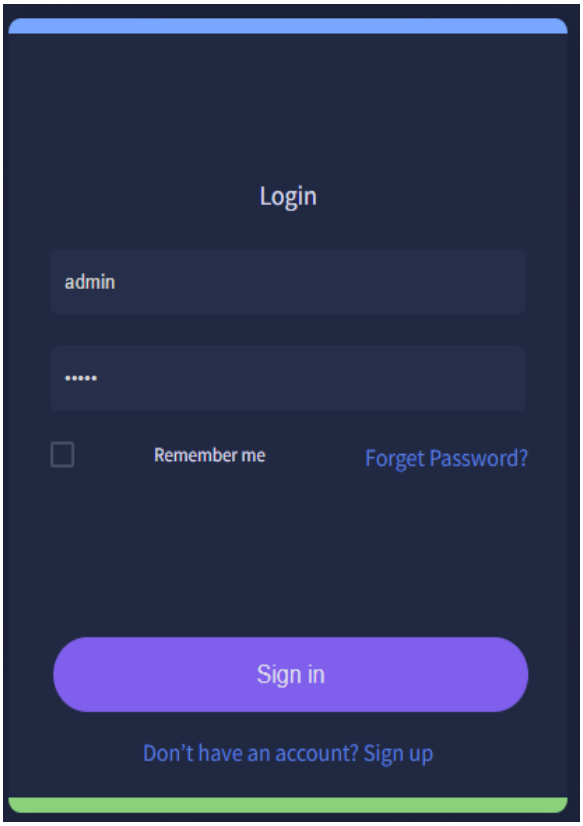

Fig 14 Login page

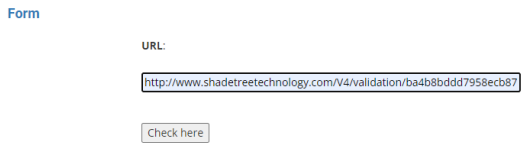

Fig 15 User input

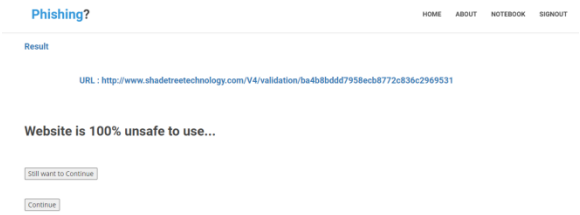

Fig 16 Predict result for given input

5. CONCLUSION

The project successfully developed Phishing Guard, a client-side defense tool utilizing advanced machine learning models, including Random Forest, Support Vector Classifier, XGBoost, and a superior stacking classifier. Phishing Guard excels at detecting and blocking malicious URLs without requiring website modifications. It leverages a diverse set of URL characteristics, such as address bar attributes, domain-based features, and HTML/Javascript properties, to enhance its accuracy and reliability. Integrated into a Flask-based front-end with SQLite-based user authentication, the tool offers a seamless and secure user experience. The project's exploration of alternative machine learning models improves predictive accuracy, ensuring Phishing Guard remains adaptable to evolving threats. This client-side focus and comprehensive feature set provide a robust and user-centric approach to online security, marking a significant step in defending against phishing attacks.

6. FUTURE SCOPE

Future iterations could explore advanced machine learning models and feature extraction techniques, continually improving the accuracy and robustness of Phishing Guard in identifying evolving phishing tactics. Incorporating mechanisms for real-time updates and adaptive learning would ensure that Phishing Guard remains effective against emerging web spoofing threats. This would involve continuous

model training based on the latest phishing data. Expanding Phishing Guard [1] to support various web browsers beyond Google Chrome would broaden its impact, providing users across different platforms with a consistent and reliable defense against phishing attacks. Integrating educational features within Phishing Guard to raise user awareness about phishing threats and safe online practices could contribute to a more resilient user community. This could include interactive tutorials or informative pop-ups. Establishing collaborative efforts with cybersecurity organizations and sharing threat intelligence could enhance Phishing Guard's effectiveness. Access to a broader dataset and collective insights would strengthen the tool's ability to recognize and thwart diverse phishing attempts.

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 naıve) 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, 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 web sites,’’ 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. Netowrks, Sep. 2008, pp. 1–6.

[16] W. Zhang, H. Lu, B. Xu, and H. Yang, ‘‘Web phishing detection based on page spatial layout similarity,’’ Informatica, vol. 37, no. 3, pp. 1–14, 2013.

[17] J. Ni, Y. Cai, G. Tang, and Y. Xie, ‘‘Collaborative filtering recommendation algorithm based on TF-IDF and user characteristics,’’ Appl. Sci., vol. 11, no. 20, p. 9554, Oct. 2021.

[18] W. Liu, X. Deng, G. Huang, and A. Y. Fu, ‘‘An antiphishing strategy based on visual similarity assessment,’’ IEEE Internet Comput., vol. 10, no. 2, pp. 58–65, Mar. 2006.

[19] A. Rusu and V. Govindaraju, ‘‘Visual CAPTCHA with handwritten image analysis,’’ in Proc. Int.

Workshop Human Interact. Proofs. Berlin, Germany: Springer, 2005, pp. 42–52.
[20] 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.
[21] P. Sornsuwit and S. Jaiyen, ‘‘A new hybrid machine learning for cybersecurity threat detection based on adaptive boosting,’’ Appl. Artif. Intell., vol. 33, no. 5, pp. 462–482, Apr. 2019.
[22] S. Kaur and S. Sharma, ‘‘Detection of phishing websites using the hybrid approach,’’ Int. J. Advance Res. Eng. Technol., vol. 3, no. 8, pp. 54–57, 2015.
[23] W. W. Cohen, ‘‘Fast effective rule induction,’’ in Machine Learning Proceedings. Amsterdam, The Netherlands: Elsevier, 1995, pp. 115–123.
[24] V. Muppavarapu, A. Rajendran, and S. K. Vasudevan, ‘‘Phishing detection using RDF and random forests,’’ Int. Arab J. Inf. Technol., vol. 15, no. 5, pp. 817–824, 2018.
[25] V. K. Nadar, B. Patel, V. Devmane, and U. Bhave, ‘‘Detection of phishing websites using machine learning approach,’’ in Proc. 2nd Global Conf. Advancement Technol. (GCAT). Rajasthan, Jaipur, India: Amity University, Oct. 2021, pp. 1–8.