AI-Powered Phishing Website Detection using Machine Learning and Browser Extension Integration

RAVULA KARTHEEK[1], T.ASHRESHTA GNAN[2], SAJID BAIG[3], S.CHAKRAVARTHY[4], T.SAI KUMAR[5]

[1] Assistant Professor, Department of CSE (DATA SCIENCE), Bapatla Engineering College: Bapatla, India.
[2, 3, 4, 5] Department of CSE (DATA SCIENCE), Bapatla Engineering College: Bapatla, India.

Corresponding Author Email: kartheekcsefaculty@gmail.com | ORCID: https://orcid.org/0009-0000-3664-3129

How to Cite this Article:

GNAN, T., BAIG, S., S.CHAKRAVARTHY, & KUMAR, T. (2026). AI-Powered Phishing Website Detection using Machine Learning and Browser Extension Integration. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.381

https://doi.org/10.55041/ijcope.v2i4.381

Abstract—

Cybercrime has many forms, and phishing is one of the most common types. There were almost 989,000 phishing incidents in the first quarter of 2024 alone! Because conventional defenses rely on blacklists to catch phishing URLs, they cannot detect phishing attacks that use new or different types of phishing URLs. This leaves millions of users at risk of having their data stolen, their identities compromised, or their money stolen as a result of phishing. This study presents a new phishing detection system called PhishGuard. PhishGuard is a browser extension that runs on Google Chrome and uses machine learning to classify each website as either a phishing website or a legitimate website in real-time. PhishGuard collects various URL-based and domain-based features of websites (e.g., URL length, special character count, HTTPS status, redirection type, etc.) and input these features into an optimized ensemble protocol that is based upon a combination of multiple classification algorithms. PhishGuard was tested against a number of established methods of phishing detection, including Logistic Regression, Decision Tree, and Support Vector Machines. The results demonstrate that PhishGuard outperformed these approaches in terms of detection accuracy. PhishGuard achieved an accuracy rating of 96.75% compared to Logistic Regression at 88.45%, Decision Tree at 91.20%, and Support Vector Machines at 90.35%. PhishGuard provides users with instant alerts after visiting a phishing website, maintains a browsing history log, and works automatically without requiring any manual user involvement. PhishGuard is an innovative, lightweight, scalable, and practical anti-phishing tool that can be deployed in any browser environment, with clear capabilities for future integration of enhanced deep learning processes and real-time threat intelligence.

Keywords— Phishing Detection, Machine Learning, Chrome Extension, Random Forest, URL Feature Extraction, Cybersecurity, Zero-Day Attacks, Real-Time Protection, Web Security, Browser-Based Security

I. INTRODUCTION

By creating a world-wide web of services using the Internet, it has changed the way in which businesses, people and governments communicate with one another. The rapid growth of the internet has also led to an increase in the number of attacks on individuals through social engineering techniques such as phishing. Phishing is defined as the act of pretending to be a reliable source with the intention of collecting personal information from the victim (Banks, 2018). It is one of the most common forms of attack across all industries; according to the APWG there were nearly 989,000 phishing attempts in the first quarter of 2024 alone [1]. This dramatic increase in phishing attacks has occurred faster than any other form of attack and has greatly outpaced the ability for traditional methods of defence against these types of attacks to keep up with them.

Phishing is usually accomplished by creating a fake website that closely resembles the visual identity and URL structure of legitimate financial service providers, retailers or government agencies. Users are tricked into clicking on links in fraudulent emails or social media posts, then being redirected to these counterfeit sites and unknowingly providing personal and financial details (login credentials, credit card numbers & PII) to the criminals operating the counterfeit site. The FBI's annual Internet Crime Complaint Center Report (IC3) alleged losses totaling greater than $10 billion annually from phishing and related fraud [2].

Malicious URLs are commonly detected by traditional security measures using a blacklist of known-neutral URLs. While using blacklists is effective to protect against known threats, they cannot be useful to protect against zero-day phishing attacks. Zero-day phishing attacks allow hackers to exploit short-lived URLs and become an attack vector against scam attacks because they do not have a record of them until after they are used. Security measures using a blacklist can be a weak form of defense for a phishing campaign when there are many short-lived URLs and if there have been updating to the design of security measures.

Heuristic and rule-based security measures are able to help reduce some of the weaknesses of blacklists by analyzing the structural attributes of HTML and URL design while issuing alerts to administrators through email or other forms of communication. In addition, heuristic and rule-based security measures still lack the ability to automatically generate false URL identification patterns on an on-going basis, therefore requiring ongoing manual adjustments, and also exhibit relatively poor performance in detecting new methods that generate obfuscations or other new infections. Thus, the research community is divided on the most appropriate way to address the movement forward in the marketplace: ML has a very strong portfolio of capabilities in that its data-based models have been shown to support discovery of very subtle, multidimensional attributes in URLs and domains, and can be retrained as new threats continue to emerge.

Even highly accurate machine learning (ML) models will have limited practical value if they cannot seamlessly integrate into a user's browsing experience. Prior research systems have generally been deployed as standalone web services, or as desktop applications, creating friction to discourage adoption [6]. In contrast, the browser extension paradigm embeds the detection process transparently into the user's existing environment by intercepting every navigation event, performing instantaneous analysis, and surfacing actionable warnings without requiring any user initiated query.

The key contributions of this paper are:

(1) An end-to-end systematic feature-engineering pipeline designed to extract 17 URL and domain level attributes that have been shown to distinguish phishing from legitimate sites

(2) A comparative evaluation of 4 commonly used classifier algorithms (Logistic Regression, Decision Tree, Random Forest, Support Vector Machine) using an open phishing dataset, demonstrating that optimal results can be obtained using the Random Forest model with an accuracy of 96.75%.

(3) A fully functional Google Chrome extension (PhishGuard) that utilizes the trained model to classify websites in real time, providing instant access alerts and a browsing history summary to users.

The structure of this article is as follows: A review of the literature will be provided in Section 2. In Section 3 we will define our data set and explain how it was prepared for analysis. Section 4 describes our system architecture; Section 5 describes how the system was developed; Section 6 provides experimental results; Section 7 discusses limitations and implications;

Section 8 proposes future research, and finally, Section 9 concludes.

II. LITERATURE REVIEW

Phishing Detection Research Evolution can be categorized into four general phases: blacklists, heuristics/rules, classical machine learning, and new methods such as deep learning and hybrids.

2.1 Blacklist and Reputation-Based Approaches

The first generation of Phishing Defense Systems used defined lists of already been identified as bad URLs and/or domains. For example, Garera et al. [7] created a method for detecting and measuring phishing using URL analysis including blacklisting. They showed through their research that many of the structural features of URLs provided enough power to tell whether or not an URL is good or bad, even if there was no previous intel on that URL as a phishing site. As an example, Google Safe Browsing [3] now performs these same functions on a large scale by preventing hundreds of millions of harmful web pages; however, one of the limitations of this approach is that it takes Google between 12-24 hours to identify a fraudulent website after it has been created so users continue to be "unprotected" during the early phases of an attack.

2.2 Heuristic and Rule Based Methods

Researchers have created heuristic engines that analyze observable features of URLs in order to mitigate the impact of zero-day blind spots. One example, developed by Fu et al. [8], is a technique for computing the visual similarity of candidate web pages against known legitimate websites using the Earth Mover's Distance, by comparing how the two websites appear when rendered. Although this is an inventive method of comparing websites' visual appearances, it would not make a good candidate for browser add-on implementations due to the high computational costs of this type of analysis. Rule-based detection mechanisms can also help detect known obfuscation patterns by flagging URLs that contain "@" symbols; excessive numbers of hyphens; numeric IP addresses; or long subdomains - however, this technique is subject to tedious manual maintenance as attackers develop new methods.

2.3 Classical Machine Learning

The Lexical Features of URL S are used in combination with host-based and content-based features for the development of competitive Logistic Regression models with less than a millisecond inference latency, by Ma et al. [9]. This demonstration was one of several early works; others (e.g., [10]) investigated support vector machines (SVMs), while others (e.g., [11]) investigated Random Forest and naïve Bayes models. These three methods consistently achieved better accuracy (accuracy) and robustness to adversarial URL crafting when combined with the ensemble of classifiers. The authors of [12] showed the Random Forest classifiers built using UCI phishing data sets achieved better than 95% accuracy, providing a strong incentive for their inclusion in this work.

2.4 Deep Learning and Hybrid Approaches

Goodfellow et al. [13] have established theoretical principles for the use of deep neural architectures for the purposes of phishing detection. Convolutional neural networks (CNNs) are able to operate using character-level embedding for URLs and long short-term memory (LSTM) networks capture patterns of sequential tokens in a URL. In both cases, accuracy of these methods was shown to be high when tested against large amounts of data [14]. Though, due to their computational requirements, deep networks cannot be used in real-time inference (e.g., as part of a browser extension) on consumer-grade computer hardware. A hybrid approach that uses cloud-based application programming interfaces (APIs) to outsource heavier processing and operations while attempting to implement a lightweight local classifier may provide a viable alternative to using a deep model independently [15], however issues of latency and privacy may arise.

2.5 Browser-Based Integration

Computer systems from both commercial and academic sources that focus on integrating browser functionality into their product line have been created. Paxson [16] discusses the need for real-time network-level attack detection, which also applies directly to analyzing URLs (Universal Resource Locators) from browsers. While many researchers are very interested in looking at browser integration, there are very few working prototypes available to evaluate because they typically only demonstrate offline capabilities. The system that we are currently using bridges the gap between offline and online capabilities; specifically, we provide a deployable Chrome Extension that will allow users to evaluate the results of our work from

the beginning step (feature extraction) to end step (user notification).

III. DATASET AND FEATURE ENGINEERING

3.1 Dataset

The experiments utilize UCI's Phishing Websites Dataset [17], a publicly available benchmark database containing thousands of labeled URLs, each represented as either phishing (class = −1) or legitimate (class = +1). With URLs representing many different URL structures, top-level domains (TLD), and characteristics, this dataset provides an excellent base for supervised classification. After eliminating any records that contained null values and subsequently removing duplicate entries from the dataset, the cleaned dataset was split into a training (80-percent) and testing (20-percent) dataset (using stratified sampling) where the class distribution for each set of data was maintained.

3.2 Feature Extraction

Feature engineering is among the most important factors in determining the success of a model for detecting phishing URLs. A collection of seventeen total features was collected from each sample URL and grouped into three feature types:

3.2.1 Lexical Features

URL Length: To obfuscate the real website's URL, phishing URLs are often exceedingly long in order to encompass links to unintended subdomains. Therefore, any URL that is more than seventy-five characters should be considered as potentially suspicious [9].

Count of Special Characters: Phishers have access to URL form field data that uses special characters (such as "@", "//", "-", and "%") to obfuscate the URL.

Subdomain Depth: Legitimately registered domains would not typically have a subdomain depth greater than two levels (i.e., a domain may have an additional two subdomains). Most phishing URLs would exceed that limit.

Presence of Suspicious Words: The occurrence of "login", "secure", "verify", or "update" as part of a URL path segment has been previously identified as a notable indicator of phishing.

3.2.2 Host-Based Features

- HTTPS Protocol: Phishing sites increasingly support HTTPS, however the lack of HTTPS support continues to be an indicator of poor quality for a phishing site. Additionally, certificate validity of <1 year raises suspicion.

- Domain Age: Phishing domains are usually registered a few days before they are used; thus a domain < 6 months old will have a heightened risk score.

- Domain Registration Length: Attackers avoid registering domains that have a long registration length; thus, domains that are registered for < 1 year are generally flagged as suspicious.

- DNS Record Availability: Non-resolvable DNS Records can signify that a domain has just been registered or is intended to be used for a temporary purpose (eg. A phishing domain).

- IP Address in the Url: Using an IP address only, within the URL, bypasses a domain's reputation and is a strong indicator that a URL was used for a phishing attack.

3.2.3 Content and Behavioural Traits

Redirect Count: Phishing sites often use multiple redirects in order to obscure their original destination.

IFrame Usage: Phishing pages often utilize hidden iframes to invisibly load malicious content.

Disabled Right Click Menu: Right click is typically disabled, to nefariously inhibit the user from inspecting the source of a page, via social engineering.

Mouseover Link Manipulation: A classic indicator of phishing is the replacement of the link displayed with a different destination URL.

These behavioral and content features were coded either as binary or continuous numeric variables. The categorical variables are one-hot-encoded. Feature matrices were normalized with min-max scaling prior to creation of the model. A Feature Importance analysis (6.3) identified the most discriminating features of URL length, HTTPS status, domain age, and redirect count.

IV. PROPOSED SYSTEM ARCHITECTURE

PhishGuard employs a multi-layered architecture composed of five logical tiers; (i). a client-side browser Chrome Extension Frontend, (ii). A URL capture and feature extraction layer, (iii). a backend preprocessing pipeline, (iv). A trained Machine Learning Inference Engine, and (v). A subsystem for presenting results and/or alerts to users. A high-level architectural description of PhishGuard is provided in the high-level architecture figure 1.

4.1 Chrome Extension Frontend

The frontend is built with HTML5, CSS3, and traditional JavaScript language and integrates a browser action popup that displays real-time classifications and the user's browsing history. Every open tab also has a content script (running within the context of that tab) which monitors the DOM navigation events in that tab and retrieves the current tab URL using the Chrome Extension Messaging API. The Chrome Extension manifest is declared with the following permissions; activeTab, storage, and host for localhost API.

4.2 Feature ExtractionAfter capturing the URL, the Chrome Extension calls the APIClient class and generates a structured payload that contains the raw URL. The backend feature extraction machine (built in Python with the Pandas and re libraries) parses the URL using the standards defined by RFC-3986 and extracts all seventeen features (described in section 3.2) from the URL. The calculated feature vectors will also be cached temporarily by the CacheManager so that duplicate URLs can be retrieved more quickly without having to make an API call at each repeat instance within a user's current session.

4.3 Inference Engine – Machine Learning

The Random Forest Classifier will use the Python pickle library for serialization to file-based persistence. The file will be loaded into memory at the time the Application Server starts. There is a Flask REST API endpoint for Inference where a feature vector is POSTed through HTTP. The API will return a binary classification with a confidence score for that feature vector. The Flask Application will run on Localhost port 3000. This ensures that all browsing data will not be sent to any external server, which is key to maintaining Privacy for Users.

4.4 Notification and Alert System

The classification results will be sent back to the Extension Popup, using the MessageHandler module, via the Inference Process. When the Model identifies a site as phishing (confidence value greater than or equal to 0.5), the Extension Popup will present a Large Red Warning Banner with a descriptive Warning Message indicating the User should not *submit Personal Information. If the site is identified as* Legitime (Valid) the Extension Popup will present a Green Indicator denoting that it is Safe to Browse. All classification events (URL, Time, and Verdict) will be persisted to the Extension's Local Storage and visible in a History Panel from the Extension Popup.

V. IMPLEMENTATION

5.1 ML Pipeline

The ML pipeline was built using Python 3.10 and two libraries - Scikit-learn and XGBoost. Data import was accomplished using Pandas' read_csv() method. Data preprocessing involved imputing missing values (mean for continuous variables and mode for binary variables); clipping any outliers at the first and ninety-ninth percentile; normalizing data to scale using min-max normalization; and partitioning the dataset into training and testing sets, with one set containing 80% of the data and the other set containing 20% of the data while maintaining the same percentage of each class in both datasets.

Four classification algorithms were used to build prediction models: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM), using a radial basis function (RBF) kernel. Hyperparameter tuning for RF was performed using RandomizedSearchCV with five-fold cross-validation across the following hyperparameter ranges: n_estimators = {100, 200, 300}; max_depth = {10, 20, None}; min_samples_split = {2, 5, 10}; max_features = {'sqrt', 'log2'}. The optimal hyperparameter settings for RF were determined to be: n_estimators=200, max_depth=20, min_samples_split=2, and max_features='sqrt' based on the mean F1-score generated from the five-fold cross-validation results.

5.2 Backend API

The Flask API server provides four endpoints, /analyze-url for URL classification, /analyze-email for email header analysis, /check-domain for domain reputation lookup, and /health for liveness monitoring purposes. The analyze-url endpoint takes a JSON body with 'url' parameter to extract features, load the pickled model, perform inference, and provide a JSON response containing classification (0 == legitimate, 1 == phishing), confidence (a floating point value between 0 and 1), and features (the extracted feature vector for audit purposes).

5.3 Chrome Extension

The chrome extension is made up of four main modules: manifest.json (the extension's permission declarations and entry points), background.js (the service worker that initializes the APIClient and MessageHandler), popup.html/popup.js (the rendered user interface), and content.js (the injected URL capture logic into the active tabs). The APIClient uses asynchronous fetch calls with a 30 second timeout and structured error handling. The CacheManager uses chrome.storage.local to persist recent URL verdicts with a configurable TTL to reduce the number of API calls to frequently accessed sites.

5.4 System and Software Specifications

Our system was tested and developed using a computer with the minimum requirements of Windows 11 (64-bit), an Intel® Core™ i5 CPU, 8 GB of RAM and a minimum of 1 GB of available hard disk space. The software required to run our system is: Python 3.10, Sci-Kit Learn (version 1.3), XGBoost 2.0, Flask (version 3.0), Pandas (version 2.1), NumPy (version 1.26) and the latest version of Google Chrome (Google Chrome 124). In addition, the file size of our model is approximately 4.2 MB and is less than the maximum file size of 10 MB permitted by Chrome extensions.

VI. EXPERIMENTAL RESULTS

6.1 Performance of Classification Algorithms

In Table 1 you can see the classification accuracy, precision, recall and f1-score for all classifiers in this study on a held out test set of data. The proposed Optimized Random Forest Classifier achieved an accuracy of 96.75%, which is a 2.65% improvement from the next best classifier (Decision Tree 91.20%) and an 8.30% improvement from Logistic Regression (88.45%).

Table 1: Classification Performance of Machine Learning Algorithms

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	88.45	87.12	86.90	87.01
Decision Tree	91.20	90.54	91.38	90.96
Random Forest	94.10	93.87	94.22	94.04
Support Vector Machine	90.35	89.80	90.11	89.95
Proposed RF (Optimized)	**96.75**	**96.42**	**97.10**	**96.76**

The enhanced performance of the optimized RF model results from its ensemble characteristcs; that is, it combines the predictions of 200 uncorrelated trees to give a total that is less variable than any singular tree's prediction alone . Comparatively low results from Logistic Regression can be seen to be a limitation of the method which cannot capture the incidence of nonlinear decision boundaries that occur in the feature space of URLs. Similarly, although SVM using a RBF kernel serves as a competitive, platform independent, benchmark, it is highly dependent on the initial selection of values for the regularization parameter (C) and kernel bandwidth (γ), both of which were not tuned exhaustively. Overfitting also appears to be a moderate issue for Decision Tree with both an accuracy result difference in training (97.80%) and testing (91.20%).

6.2 Analysis of Performance Metrics

In the context of phishing detection, the recall measure is more important than just accuracy: A false negative (where a phishing site is perceived as legitimate), would directly put users in harm's way; a false positive (where a legitimate site is inadvertently marked as a fraud) would not put users at risk, but would merely give them inconvenience. The proposed RF Model has a recall of 97.10%, meaning only 2.9% of phishing sites go undetected. This is an improvement over the 86.90% Recall for Logistic Regression. A 96.42% Resolution also ensures that only 3.6% of the alerts generated for safe sites would

be spurious, which helps to maintain user confidence in the system.

Lastly, with a F1-Score of 96.76%, this model achieves an optimal balance between Precision and Recall and should be considered for implementation in environments where minimization of both false alarm fatigue and the risk of a security breach should be prioritized.

6.3 Feature Importance Analysis

Seven key features are ranked based on how they contribute towards improving the Random Forest model's accuracy. The features include URL length, number of dots in URL, domain age, HTTPS token presence, redirect count, presence of "@" symbol and IP address in URL. The combined importance of these seven features represents approximately 71% of the total predicted information gain for the Random Forest model confirming existing literature support [9, 11]. Features contributing lesser towards overall model accuracy (i.e., right click disablement and iframe usage) also remained in the final Random Forest model due to their relatively modest contributions and lack of increased inference time when added to the random forest classifier.

6.4 Performance of the Real-Time System

Five hundred test subject navigation events were utilized to calculate average end-to-end latency values (from URL capture to alert presentation). The 143 ms (SD = 28 ms) average latency had a 95th percentile of 187 ms which should be imperceptible to users and not significantly impede their browsing experience. Additionally, because of the CacheManager's ability to reduce the number of API calls made per repeat URL visit in simulated browsing sessions by 38%, responsiveness was also improved.

VII. DISCUSSION

7.1 Comparison to Current Systems

PhishGuard provides two additional capabilities when compared with current systems, such as Google Safe Browsing, which rely on purely blacklist methods of protection (3). First, PhishGuard is able to identify and classify both zero-day phishing URLs, as they have not yet been identified in any threat database, and operate entirely locally, resulting in no delay (latency) due to the need to query a cloud API for data on site indexing and/or lack of privacy from those queries. Second, PhishGuard incorporates end-to-end browser integration versus the offline way that other machine learning prototypes have been tested (e.g., (9) (11)).

7.2 Limitations of Current System

The current system evaluates only URL-structural features and domain-metadata and does not evaluate the rendered webpage, or scripts that are embedded in the rendered webpage, as well as the visual similarities to those pages identified as legitimate. This lack of ability to evaluate the rendered version of a webpage, in conjunction with the fact that the phishing websites employed by sophisticated phishing attacks continue to use valid, registered HTTPS domains, and have few URL anomalies, may lead to lower confidence in the classification of those sites as being phishing. The current system relies on architecture using a local flask server; thus, the users' machine must be running in order for the system to work, which is a friction-point issue that will need to be addressed in production deployments by bundling the inference engine as either a native messaging host or a WASM module in the extension itself. Although this dataset has become an industry-standard benchmark dataset, it does not incorporate all the contemporary phishing tactics being employed today, nor does it incorporate all possible IDN homograph attacks and all types of typosquatting, nor does it provide for sufficient sample sizes of identical privatively registered domains having been registered within a given time period and all domains containing only a single letter in TLD and/ or length of the domain as a whole.

7.3 Ethical Considerations

All URL data used for training is composed entirely of publicly available datasets from the UCI repository [17] and does not include any browsing data from actual users. The extension saves user browsing history in user's local chrome.storage.local namespace; it cannot be accessed by any remote servers. Users maintain full control of their browser history via the ability to clear it from their local chrome.storage.local. Both sets of design decisions have been made in accordance with the concept of privacy by design [18].

VIII. FUTURE WORK

Many options are available to upgrade the technology. Firstly, Google's Safe Browsing API [3] can give our system more detail when used as an extra source to detect bad URLs. The Machine Learning Model provides safety from zero-day attacks and, by combining the API and the Machine Learning Model, we can create a complete defence method. Secondly, character-based models (CNN or LSTM) have shown the potential for use directly against URL character strings as opposed to extracting URL features, thus removing the monthly engineering that would be required to implement our programme. Thirdly, the extension can be extended to work across many browsers (Firefox, Microsoft Edge, Safari) through the use of Manifest V3 compliant APIs. Fourthly, by implementing an anonymous Federated Learning Framework whereby anonymised updates of the model from different browsers using the extension are sent to, and aggregated by, our service provider without sending any raw data relating to your browsing behavior, the model will be able to learn from new phishing activities while maintaining user privacy. Finally, by adding user input into the cycle (e.g., who will flag false positives and negatives), we can leverage that feedback as an input into periodic training of the model.

IX. CONCLUSION

This study introduced PhishGuard, a phishing detection system that uses machine learning to provide real-time, browser-based protection via a Google Chrome extension. The technology combines a sophisticated set of 17 features derived from URL/domain data using an optimized Random Forest Classifier (achieving 96.75% accuracy, 96.42% precision, 97.1% recall and 96.76% F1-score on a standard phishing benchmark) and surpasses the performance of traditional methods (Logistic Regression, Decision Trees, SVM) for phishing detection. The end-to-end architecture presented in the study is transparent (capturing URLs), includes feature extraction, performs machine learning inference, and provides near concurrent alerting to users about phishing attacks. The overall latency associated with using PhishGuard is less than the latency threshold of 143 ms (the average latency experienced during interactive web browsing). PhishGuard can detect zero-day attacks on a proactive basis because it uses generalised patterns rather than cataloguing URLs that have already been found to be malicious (addressing the limitations of using only a blacklist defence approach). Given the increasing volume and sophistication of phishing attacks, practical and deployable solutions such as PhishGuard will be critical elements in a layered cyber-security defence strategy.

REFERENCES

[1] Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report, Q1 2024," APWG, 2024. [Online]. Available: https://apwg.org/trendsreports/

[2] Federal Bureau of Investigation, "Internet Crime Report 2023," FBI Internet Crime Complaint Center (IC3), Washington, D.C., 2024.

[3] Google, "Google Safe Browsing API Documentation," Google LLC, 2024. [Online]. Available: https://developers.google.com/safe-browsing

[4] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A Framework for Detection and Measurement of Phishing Attacks," in Proc. ACM Workshop on Recurring Malcode (WORM), Alexandria, VA, USA, 2007, pp. 1–8.

[5] B. Zhu, B. Kang, J. Li, and J. Ma, "Phishing Detection Using URL-Based Heuristics with Cross-Site Validation," IEEE Access, vol. 9, pp. 28574–28587, 2021.

[6] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting Phishing Websites Based on Self-Structuring Neural Network," Neural Computing and Applications, vol. 25, no. 2, pp. 443–458, 2014.

[7] S. Garera, N. Provos, M. Chew, and A. Rubin, "A Framework for Detection and Measurement of Phishing Attacks," ACM Workshop on Recurring Malcode, 2007.

[8] A. Y. Fu, W. Liu, and X. Deng, "Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance," IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 4, pp. 301–311, 2006.

[9] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Learning to Detect Malicious URLs," ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 3, pp. 1–24, 2011.

[10] I. Fette, N. Sadeh, and A. Tomasic, "Learning to Detect Phishing Emails," in Proc. 16th International World Wide Web Conference (WWW), Banff, AB, Canada, 2007, pp. 649–656.

[11] L. Tan, C. J. Lim, and A. B. J. Teoh, "PhishPrint: Evaluating Phishing Website Detection Mechanisms," Computers & Security, vol. 112, 2022, Art. no. 102494.

[12] C. L. Tan, K. L. Chiew, and K. Wong, "PhishWHO: Phishing Webpage Detection via Identity Keywords Extraction and Target Domain Name Finder," Decision Support Systems, vol. 88, pp. 18–27, 2016.

[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[14] W. Yang, W. Zuo, and B. Cui, "Detecting Malicious URLs via a Keyword-Based Convolutional Gated-Recurrent-Unit Neural Model," IEEE Access, vol. 7, pp. 29891–29900, 2019.

[15] X. Han and Y. Xu, "Hybrid Phishing Detection with Lightweight Local Classifier and Cloud-Based Intelligence," in Proc. IEEE International Conference on Cybersecurity and Cloud Computing (CSCloud), 2022, pp. 45–52.

[16] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Computer Networks, vol. 31, no. 23–24, pp. 2435–2463, 1999.

[17] R. Mohammad, F. Thabtah, and L. McCluskey, "Phishing Websites Dataset," UCI Machine Learning Repository, 2015. [Online]. Available: https://archive.ics.uci.edu

[18] A. Cavoukian, "Privacy by Design: The 7 Foundational Principles," Information and Privacy Commissioner of Ontario, Canada, 2009.