International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.107, May 2026

Runtime Phishing URL Detection using Heuristics and Machine Learning

Md. Aminul Hoque Shamim
Department of ICT
Bangladesh University of Professionals
Dhaka - 1216, Bangladesh

ABSTRACT

As the internet plays an increasingly integral role in our daily lives, the menace of phishing attacks has become a pressing concern, threatening the security and privacy of users worldwide. Phishing URLs, in particular, represent a significant challenge due to their deceptive nature, effectively mimicking legitimate websites to lure unsuspecting victims into divulging sensitive information. This thesis delves into the development and implementation of an advanced phishing URL detection system using state-of-the-art machine learning techniques. The research focuses on harnessing the power of supervised learning algorithms, such as Random Forest, Support Vector Machines, and Gradient Boosting algorithms, to analyze diverse features extracted from URLs and accurately classify them as legitimate or phishing attempts. The dataset used in this work comprises a large collection of real-world phishing URLs along with legitimate ones, curated from the Kaggle machine learning repository. The proposed system demonstrates promising results, achieving high accuracy, precision, and recall rates in distinguishing between genuine and malicious URLs. The study contributes to the field of cybersecurity by providing an efficient and robust solution to combat phishing threats and protect users from falling victim to cybercrime. Furthermore, it opens up avenues for further research in the realm of machine learning-based cybersecurity applications, emphasizing the significance of proactive measures to safeguard digital ecosystems from emerging cyber threats.

General Terms

Phishing, Security and Privacy, Machine Learning

Keywords

Phishing URL, Machine Learning, Random Forest, Support Vector Machines, Gradient Boosting

1. INTRODUCTION

1.1 Background and Motivation

The rapid growth of the digital services and the convenient integration of the digital into normalcy has created unparalleled convenience and connectivity to the billions of users around the world. However, this computer revolution has expanded the reach of the attack by the high-level cyber threat as well. Among them, one of the most common, ongoing, and deceitful problems in the contemporary cybersecurity space is phishing attacks [1]. Phishing is a socio-technical fraud, which means that it involves manipulative methods, in this case, deceitful Uniform Resource Locators (URLs) that appear to belong to legitimate domains to entice unwary users to share sensitive information (logins, financial data and personally identifiable information) with phishers.

Although there has been a major improvement in the security infrastructure, phishing URL detection and alleviation are a challenging activity. The main reason behind this challenge is largely connected with the fact that the sphere of phishing attacks evolves very fast, and the amount of new attacks offered on a daily basis is staggering [2]. The classical means of defense are the rule-based filtering and signature-based blacklisting which are practically reactive. Any form of tracking and blacklisting of similar malicious URLs (manually or by crowdsourcing) is not only time consuming and subject to human error, but also presents the user with a highly vulnerable position prior to an official classification of a new phishing domain. The traditional practices, in turn, cannot be used anymore to counter the modern threats as attackers are becoming increasingly more evolved in terms of evasion strategies.

This constantly growing arms race needs a paradigm shift on the reactive to proactive and automated detection strategies. A good defense mechanism must be capable of distinguishing amongst good and bad URLs, with high precision and low latency. Machine learning (ML) with its own inherent ability to discover complex, nonlinear tendencies in large-scale data sets and change in response to the emergent threat vectors is a highly promising solution. Ml models require optimization however to be able to be used in real world, time sensitive situations. Known heuristics, i.e. domain specific guidelines and structural anomalies, which tend to be typical of fraudulent relationships, and a powerful machine learning algorithm may be regarded as a synergistic strategy that will raise the detection rate and decrease the cost of the computation effort.

Thus, the main goal of the given study is to plan, deploy, and offer a sound, hybrid phishing URL detection platform. Through the application of the power of supervised learning algorithms enhanced by heuristic analysis, it is this research that will present a model that is capable of analyzing granular URL features with the aim of finding subtle indicators of compromise, thus identifying previously unknown, advanced attacks. Importantly, to handle the immediate threat on the end-user, this system will act like a runtime detection engine, and will offer on the fly analysis and warning generation.

1.2 Research Contributions

In order to fill the research gaps that currently exist in the literature as well as meet the research objectives, this study will contribute to the body of research in the following aspects:

—The study suggest a new, automated detection algorithm that provides high-quality phishing URL classification using the machine learning algorithms.

—It performs an extensive feature engineering study to identify the most topical structural and lexical features that are necessary to classify legitimate and phishing URLs.

—It presents a hybrid framework of detection, which combines existing URL heuristics and a machine learning pipeline, and has better detection rates in comparison with stand alone methods.

—It creates and tests a runtime deployment engine which can process URLs in real-time, which proves it to be useful in protection of users in real-time and warning generation.

2. LITERATURE REVIEW

To become successful in counteracting the progressive threat of phishing, the understanding of the dynamics of URL manipulation and the development of the policy of detecting it is needed. This section will provide a background of the characteristics of phishing URLs, will provide a cursory overview of the machine learning algorithms on the field and lastly criticize the literature as it currently stands in the field of phishing in various detection paradigms.

2.1 Anatomy of Phishing URLs and Evasion Techniques

Phishing URL is a spoofed Web address which is done so that it looks like a legitimate domain with the aim of defrauding and looting of sensitive information. There are many obfuscation methods used by attackers in order to circumvent the traditional security filters and take advantage of human perception. The most frequent techniques of manipulation are:

—**Typo-squatting and Misspellings:** Replacing characters with similar looking ones (e.g. replacing the lowercase letter l with the uppercase letter I in the web address www.netfIix.com).

—**Subdomain Manipulation:** Adding misleading subdomains to a malicious root domain in an attempt to generate a false secure feeling (e.g. www.paypal.verify-account.com).

—**Homograph Attacks:** With the help of characters of other scripts (e.g. Cyrillic), the visual appearance of the URL is identical to that of the target domain.

—**Obfuscation through Shorteners:** Use of URL shortening services (e.g. bit.ly) to cover the real destination of the malicious link.

The drawbacks of reactive and signature-based defenses have triggered the use of URL feature-based detection. This proactive approach mined, semantically actuated, lexical and structural characteristics, such as length of domain, use of hyphen or suspicious keywords and rendered it easy to categorize lightly and in real-time, without attending to what is on the sites. Furthermore, the use of these detection engines as browser extensions (through the background and contents scripts) has been viewed as a good method towards runtime interception and real-time warning generation.

2.2 Review of Phishing Detection Methodologies

The latest attempts at automating phishing detection can be generally divided into the heuristic/list-based, visual similarity, machine learning (ML), and deep learning.

2.2.1 Heuristic and List-Based Approaches. The heuristic approaches are based on the predetermined rules that are guided by the knowledge about the domain to apply in the process of flagging suspicious URLs and the list-based approaches are based on whitelists or blacklists. Lexical characteristics were used to detect spam and phishing URL by Kumar et al. [5], and they achieved 97.7 percent accuracy with a Random Forest classifier; however, they were not tested on separate training and test data. Similarly, Gupta et al. [6] found that a lightweight heuristic model written in Random Forest achieved a high accuracy of 99.57 percent but the model can only be used in a small range of conditions due to the few featured elements (9 lexical features). Almeida and Westphall [7] suggested a heuristic string-matching approach with SVM (98.05% accuracy), however, they observed that standalone heuristics were not able to match dynamic evasion strategies. Listing methods, such as the whitelist method proposed by Azeezet et al. [15], are highly specific but cannot scale well and have false-positive rates in case of a user access to recently created domains of legal users. Barraclough et al. [23] also achieved the 99.33percent accuracy rate using an heuristics based ML classifier (PART algorithm), but the sets of rules need to be updated manually to maintain them.

2.2.2 Visual Similarity Approaches. The visual similarity methods detect phishing by comparing screenshots, logos, and layouts of a suspicious webpage to a database of legitimate targets. In Ding et al. [8], a visual method of search with an accuracy of 98.90% was suggested; this time around, the search-engine based approach adopted in this scenario is really high in the terms of latency and hence not viable in the real-time form of implementation. Van Dooremaal et al. [11] and Li et al. [12] also used visual and DOM-based features, which obtained high accuracies (99.66 and 98.60, respectively) but their methods are computationally expensive and relied on the limited datasets. Rao et al. [9] proposed a light-weight visual scheme, that is, word embeddings, and Wang et al. [10] implemented Optical Character Recognition (OCR) to the cell phones. The visual similarity techniques are well-suited in zero-day attacks, however, have high computational costs, and fail when directed at non-whitelist domains [9].

2.2.3 Machine Learning and Deep Learning Approaches. The use of high-end machine learning has become the standard of scalable phishing detection, such as Decision Trees, Support Vector Machines (SVM), as well as ensemble-based techniques, such as Random Forest and XGBoost. Hannousse and Yahiouche [16] demonstrated that compared to traditional classifiers, ensemble models i.e. random forest is a superior model when applied to hybrid features. Stobbs et al. [17] stressed on the significance of hyperparameter optimization that they used in their experiment and minimized the error to 99.33 per cent with the assistance of Random Forest optimized with the assistance of Genetic Algorithms. Nevertheless, there are cases where ML implementations cannot handle imbalanced datasets which are complex like Geyik et al. [18] and Anupam and Kar [19] who obtained much lower accuracies of 83.0% and 90.38, respectively.
Deep Learning (DL) architectures have been thought of recently as far as automated feature extraction is concerned. A case in point is the deep autoencoder suggested by Bu and Cho [20], which can classify the URL strings of characters with an accuracy of 97.34%.

Korkmaz et al. [21] employed the Convolutional Neural Networks (CNN) but the accuracy was quite low (88.90). As was shown by Feng and Yue [22], Recurrent Neural Networks (RNN) was capable of reaching detection accuracy of 99.50 with purely lexical features. The absence of human-engineered functionality in DL models makes them extremely problematic to execute on the client-side although their black-box design, astronomical computation requirements and long inference times are highly problematic to client-side execution.

2.3 Summary and Research Gaps

The Table 1 presents the important contributions, methods, and limitations of the reviewed literature.
Although the existing literature proves a high level of classification accuracy, there are still gaps. Deep learning and visual similarity methods present unacceptable latency of on-the-fly detection. On the other hand, very precise heuristic models may use third-party services or content-on-demand, which violates the necessity of a fast and standalone response. Thus, the present literature is quite evident in the need to have a lightweight, hybrid system that integrates rapid heuristic extraction with a powerful machine learning, which client-side runtime execution and third-party API-free deployment.

3. RESEARCH METHODOLOGY

This study presents a hybrid defense-in-depth architecture to overcome the shortcomings of standalone phishing detection methods reported in the literature. The main goal is to distinguish phishing URLs with high accuracy but at the same time low latency needed to provide real-time user protection. To do this, this study propose a powerful runtime detection engine implemented as a browser extension, that combines machine learning, heuristic feature engineering, and third-party threat intelligence.

3.1 System Architecture Overview

The system proposed is a non-blocking, client-side runtime engine. On attempting to access a new Uniform Resource Locator (URL), the system captures the request and at the same time checks the URL using three different subsystems, which are running in parallel before displaying the page.
The building is made up of three main components which are as follows:

—**Machine Learning Classification Engine:** A lightweight, pre-trained predictive model, which analyzes the URL against learned patterns.

—**Heuristic Feature Extraction Engine:** A deterministic program which inspects the structural and lexical characteristics of the URL to raise suspicious anomalies.

—**External Threat Intelligence Engine:** A cloud-based validation element that makes use of Google Safe Browsing API v4.

The system has a stringent, conservative decision making matrix to maximize user security. A URL is only considered to be benign and thus allowed to pass on when all the three subsystems by themselves classify the URL as benign. When any of the components raise a red flag on the URL as suspicious or malicious, the system automatically creates a runtime warning, which blocks access.
The study uses machine learning technique to classify a phishing URL. For that the approach is to train model with a state of the art data set and then evaluate the trained model to classify unseen data to positive (phishing url) or negative (non phishing url) cases. Apart from that, the study apply heuristics based feature engineering to further enhance the accuracy of the proposed approach. Finally, a phishing URL detection strategy has been embedded as a flexible browser extension to facilitate runtime warning generation as suspicious URLs are accessed.

3.2 Dataset Description and Preprocessing

Another important part of the planned machine learning pipeline is the dataset that will be used to train and evaluate the machine learning in offline mode. The current study makes use of the publicly available Phishing Websites Dataset on the UCI Machine Learning Repository [28].

3.2.1 Dataset Characteristics. The dataset is balanced, a mix of legitimate and phishing URLs, so the model will be trained on a representative sample with no significant bias due to a class imbalance. It has 11055 unique examples, each of which has 31 different, pre-extracted features. They are essentially categorical features, which can take binary or ternary values depending on the indicator.

3.2.2 Feature Categorization. The characteristics are grouped into four different categories, as shown in Table 2. This multidimensional solution will be used to make sure that the model can capture the different attack vectors, including lexical obfuscation and domain level anomalies.

3.2.3 Data Preprocessing. In order to achieve a quality, normalized representation of the data and achieve the best performance of the classifier, a tight-fitting preprocessing pipeline was deployed. The first step involved cleaning the data by parsing all the URLs to their constituent parts and removing invalid or un-parcelable entries, which ended up in a final dataset of 11,054 high-fidelity instances.
An exploratory data analysis (EDA) was performed to determine the basic statistical characteristics of the features. The following preprocessing transformations were performed on the dataset:

—**Missing Data Handling:** Scaling Structural integrity.

—**Correlation Analysis:** This is the evaluation of class label against other feature correlations to remove the irrelevant or very collinear features that may cause noise or overfitting.

—**Standard Scaling and Normalization:** Scaling the feature vectors to a standard scale to make the training of algorithms faster.

Lastly, to avoid being biased towards particular data qualities and to make sure that the model can be generalized to unknown threats, the processed dataset was split into disjointed training and testing sets by a random sampling method. A ratio of 80:20 was taken: 80 percent of the cases were used to train the models, and the remaining 20 percent of cases were strictly held as a hold-out test set to evaluate the performance of the models in an unbiased manner.

3.3 Heuristic Feature Extraction Engine

The initial step of the detection approachology is based on deterministic heuristic engineering. Machine learning models are effective, but they may not pick edge cases that break basic URL structuring rules. This subsystem is a fast, light filter.
This engine interprets the URL upon encountering it and breaks down the string into its components (protocol, domain, path, query parameters) and retrieves a set of predefined lexical features. These features are selected in particular due to typical obfuscation methods used by attackers, such as:

Table 1. Comparative Summary of Phishing Detection Literature

Ref.	Authors	Domain	Methodology	Key Contribution
[1]	Kumar et al. (2018)	Phishing	RF, MLP	Achieved 97.7% accuracy using hybrid ML models; however, limited evaluation with only two classifiers and lack of diverse datasets reduced generalizability.
[2]	Gupta et al. (2021)	Phishing	Random Forest	Reported 99.57% accuracy but relied on limited URL-based features, raising concerns about robustness across diverse attack patterns.
[3]	Almeida & Westphall (2020)	Phishing	SVM	Twin SVM achieved 98.05% accuracy, though performance dropped significantly (73.3%) under varying conditions, indicating instability.
[4]	Ding et al. (2019)	Phishing	Logistic Regression	Obtained 98.90% accuracy using combined visual and heuristic features but depended heavily on third-party search engines, increasing latency.
[5]	Rao et al. (2020)	Phishing	LR, SVM, RF, DT, XGBoost	Achieved accuracy between 97.66% and 98.72% using multiple classifiers; however, dataset size was limited and computational cost was high.
[6]	Wang et al. (2020)	Phishing	OCR-based Detection	Effectively detected phishing through visual similarity, though highly dependent on device and language constraints.
[7]	van Dooremaal et al. (2021)	Phishing	Logistic Regression	Achieved 99.66% accuracy; however, reliance on search engines caused fluctuating performance over time.
[8]	Li et al. (2019)	Phishing	GBDT, XGBoost, LightGBM	Stacking models achieved 98.60% accuracy, but dataset size was relatively small (2,000 webpages).
[9]	Barlow et al. (2020)	Phishing	Neural Network	Reported 94.16% accuracy, though evaluation was limited to a single neural network architecture.
[10]	Maroofi et al. (2020)	Phishing	RF, Logistic Regression	Achieved 97.00% accuracy using RF but relied heavily on third-party features, increasing processing time.
[11]	Azeez et al. (2021)	Phishing	Whitelist + Visual	Reported 96.17% accuracy; however, experiments were conducted on a very small dataset, limiting reliability.
[12]	Hannousse & Yahiouche (2021)	Phishing	SVM, DT, LR, RF, NB	Hybrid feature-based RF achieved best performance (96.61%), though content-based features were unsuitable for real-time detection.
[13]	Stobbs et al. (2020)	Phishing	Random Forest	Achieved 99.33% accuracy with optimized hyperparameters but required computationally expensive optimization methods.
[14]	Geyik et al. (2021)	Phishing	LR, NN, SVM, DT, NB, RF	Random Forest achieved highest accuracy (83.0%), but overall performance was lower than existing state-of-the-art approaches.
[15]	Anupam & Kar (2021)	Phishing	GWO + SVM	Grey Wolf Optimizer improved performance to 90.38%, though still lower than competing methods on the same datasets.
[16]	Bu & Cho (2021)	Phishing	RNN	Achieved 97.34% accuracy but focused mainly on character-level URL features, limiting feature diversity.
[17]	Korkmaz et al. (2021)	Phishing	CNN	Reported 88.90% accuracy, indicating comparatively weaker performance among deep learning models.
[18]	Feng & Yue (2020)	Phishing	RNN	Achieved 99.50% accuracy but evaluated only a single RNN model with high computational cost.
[19]	Barraclough et al. (2021)	Phishing	PART, J48, JRip, ANFIS	PART algorithm achieved 99.33% accuracy, though reported a relatively higher error rate and model complexity.

—**Domain Anomalies:** Domain length is too long, there are IP addresses and not domain names, and multiple sub domains are used to resemble legitimate sites.

—**Character Obfuscation:** The existence of the symbol of the at mark (in many cases to cover the actual destination) as well as the use of too many hyphens (-) in the domain and the use of non-standard port numbers.

—**Keyword Matching:** Sensitive or time-sensitive keywords, such as secure, login, verify, update, etc., in the URL path or subdomains.

3.4 Machine Learning Pipeline and Transpilation

The machine learning subsystem is the main part of the prediction ability. A two-stage approach was applied: the offline training stage and the online inference in the run stage.

3.4.1 Offline Model Training. The preprocessed UCI data is used to train the model offline. Trained different supervised machine learning classifiers to determine the best algorithm to achieve a compromise between high classification accuracy and low computation complexity, an essential aspect of making the computation run smoothly in a browser.

3.4.2 Model Translation and Deployment. One major technical issue with Python-based machine learning models (e.g., models

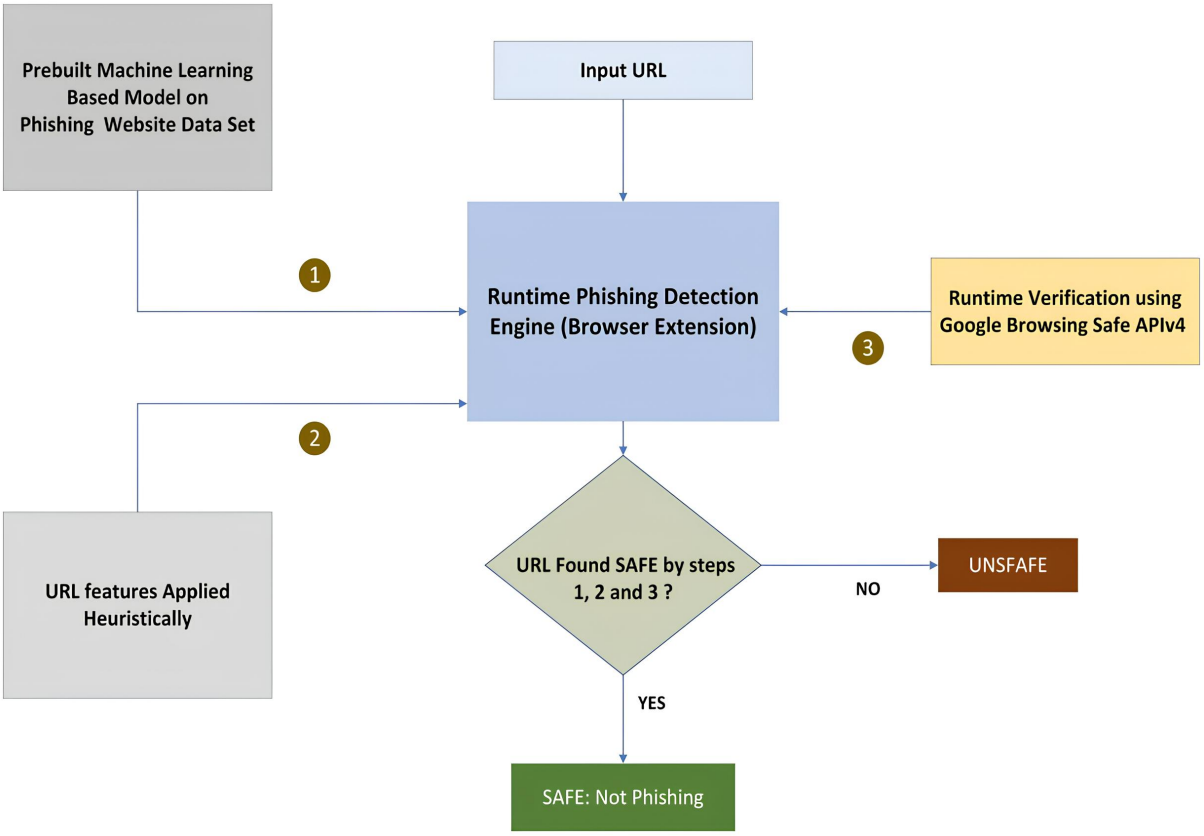

Fig. 1. High-level architectural overview of the tri-layered runtime phishing detection system, illustrating the parallel execution of ML, heuristics, and API components.

Table 2. Taxonomy of Features in the UCI Phishing Dataset

Category	Description	Key Features
Address Bar	Lexical properties of the URL string.	IP Address, URL Length, Shortening Service, "@" Symbol, Double Slash Redirection, Prefix/Suffix, Having Subdomain, SSL State.
Abnormal	Discrepancies between URL and content.	Request URL, URL of Anchor, Links in Meta/Script/Link tags, Server Form Handler (SFH), Abnormal URL.
HTML/JS	Client-side scripting anomalies.	Website Forwarding, Status Bar Customization, Disabling Right Click, Using Pop-up Windows, IFrame Redirection.
Domain-Based	Network and registration metrics.	Age of Domain, DNS Record, Website Traffic, PageRank, Google Index, Statistical Reports.

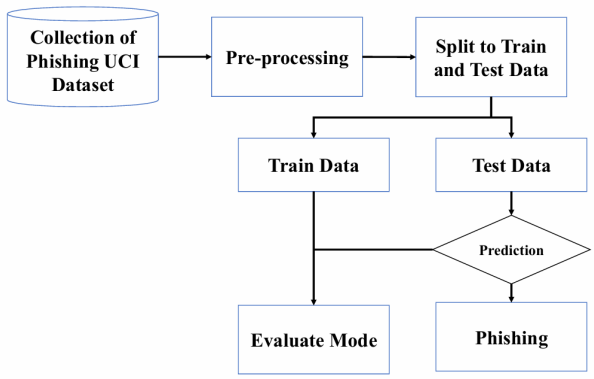

Fig. 2. Workflow of Machine Learning Based Phishing Detection

written in Scikit-learn) is the hard sandboxing and security restrictions that do not allow Python code to be executed.

To overcome this drawback without reducing the predictive ability of the model, a transpiration pipeline was built. After training and validating the best machine learning model in Python, transpiler tools and scikit-learn.js integration framework was applied. This converts the generated model weights, decision trees and evaluation logic into lightweight, native JavaScript. Accordingly, the model evaluation code is executed safely and natively in the JavaScript environment of the browser extension, and does not have to transmit user browsing information to an endpoint server to perform an ML inference.

3.5 External Threat Intelligence Integration

The system combines the Google Safe Browsing API version 4 to supplement the predictive and heuristic models with real-time, world threat intelligence.
This subsystem works by sending an asynchronous POST HTTP request to the API endpoint, with the target URL contained in a JSON payload, and a secure API key. The Google Safe Browsing service matches the URL with continuously updated lists of web resources that are unsafe. As long as the API response data shows a match in its threat data, the URL is flagged as certain. This serves as an important safety net of very active and internationally famous phishing campaigns which can go through structural heuristics.

3.6 Hybrid Detection Logic (Conservative Approach)

These three layers are combined into one decision matrix and this is the core innovation. This is formalized in Algorithm 1.

Algorithm 1 Hybrid Phishing Detection Logic

Input: URL U
Output: Classification $C \in \{Safe, Phishing\}$
Extract features F_h for Heuristic Engine
Extract features F_{ml} for Machine Learning Engine
$R_h \leftarrow Evaluate\_Heuristics(F_h)$
$R_{ml} \leftarrow Evaluate\_ML\_Model(F_{ml})$
$R_{api} \leftarrow Query\_Google\_Safe\_API(U)$
if $R_h = Safe$ **and** $R_{ml} = Safe$ **and** $R_{api} = Safe$ **then**
    $C \leftarrow Safe$
else
    $C \leftarrow Phishing$ {Generate Runtime Warning}
end if
Return C

3.7 Runtime Deployment Workflow

The final step of the proposed methodology will be its implementation in the form of a single browser extension. The extension silently runs in the background, performing the workflow shown in Figure 2.
The workflow makes sure that the performance overhead is kept down. The Heuristic Checker and the translated JavaScript ML Model check the URL locally on the client-side machine with virtually no latency. At the same time, a request to the Google Safe API is performed asynchronously. These results are collected by the browser extension. When the strict consensus logic is met (all systems reply safe), the user is automatically redirected to the webpage. Otherwise, the content script extension will insert a red flag overlay into the Document Object Model (DOM), disabling the user to use the potentially bad site.

3.8 Performance Evaluation Metrics

To evaluate the effectiveness of the proposed methodology, four standard classification metrics were utilized based on the Confusion Matrix, where TP (True Positive) and TN (True Negative) represent correct classifications, and FP (False Positive) and FN (False Negative) represent errors.

(1) **Accuracy:** The ratio of correctly predicted observations to total observations.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

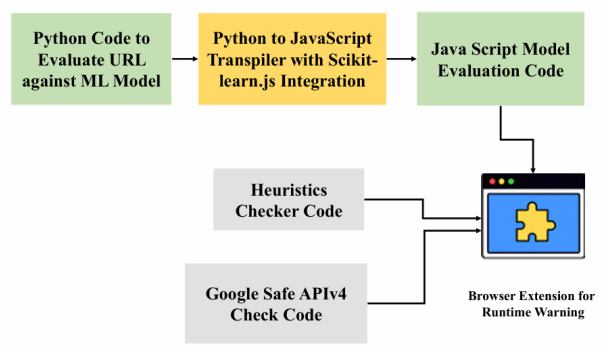

Fig. 3. Workflow of deployed browser extension. The figure shows how the offline Python ML model can be transpiled into JavaScript, and then integrated with the heuristic checker and Google Safe Browsing API to give run-time warnings.

(2) **Precision:** The ratio of correctly predicted positive observations to the total predicted positives.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

(3) **Recall (Sensitivity):** The ratio of correctly predicted positive observations to all observations in the actual class.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

(4) **F1-Score:** The weighted average of Precision and Recall.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

4. RESULTS AND DISCUSSION

4.1 Classifier Performance

Four machine learning classifiers, including XGBoost, Random Forest, Support Vector Machine (SVM), and Decision Tree, were tested on the test set in terms of accuracy, F1-score, recall and precision. Table 3 summarizes the results.
XGBoost was the best of all classifiers with the best overall performance, accuracy of 0.974, F1-score of 0.977, recall of 0.994, and precision of 0.986. The high recall rate shows that the model was able to detect the huge majority of real phishing URLs in the dataset, reducing false negatives, which is a crucial factor in security-sensitive applications. The F1-score of 0.977 indicates that there is a balanced trade-off between precision and recall, which proves that the model does not put a disproportionate emphasis on either of the metrics.
The second-ranked model was the Random Forest with an accuracy of 0.967 and a recall of 0.993, which proves that the ensemble-based models are typically very well-suited to phishing URL classification. SVM and Decision Tree had accuracies of 0.964 and 0.960 respectively. It is interesting to note that the Decision Tree resulted in the highest precision (0.993) of all the classifiers, but with a slightly lower F1-score, indicating a bias towards fewer false positives at the expense of a relatively more false negatives compared to XGBoost. By and large, the experimental findings demonstrate the fact that the proposed machine learning-based phishing URL detection system can be reliably used to differentiate between legitimate and phishing URLs with all of the considered metrics.

Table 3. Phishing URL Classifier Performance

ML Model	Accuracy	F1 Score	Recall	Precision
XGBoost Classifier	0.974	0.977	0.994	0.986
Random Forest	0.967	0.971	0.993	0.990
Support Vector Machine	0.964	0.968	0.980	0.965
Decision Tree	0.960	0.964	0.991	0.993

4.2 Comparison with State-of-the-Art

Table 4 compares the performance of the proposed system to four representative phishing URL detection studies in the existing literature to put the performance of the proposed system in perspective. A study by Sahingoz et al. [20] (2019) suggested a phishing detection system based on NLP-based URL features and a Random Forest classifier, which had 0.979 accuracy. Although their method proved to be highly accurate, F1-score, recall, and precision were not explicitly stated, and their system only works on URL-level features but cannot detect features at runtime.
In this comparison, [21] reported the highest accuracy (0.993), F1-score (0.992), recall (0.993), and precision (0.992) in a deep learning ensemble that consisted of a Convolutional Neural Network (CNN) and a Random Forest. Their model however makes use of the URL and HTML content features, and as such, it consumes much more computational resources as well as architectural complexity. Importantly, this method fails to facilitate the detection of runtime URL, and this restricts its practicality in real-time deployment scenarios.
In the study by Fares et al. [22] (2024), SVM was used to identify email phishing attacks with URL and content-based features and achieved a high accuracy of 0.976. Their work however is based on the email domain and not the general web URL classification and they did not report recall and precision values to make a direct comparison. Nugroho and Suryono [23] (2025) used Gradient Boosting on a URL dataset obtained through Kaggle with an accuracy of 0.980, F1-score of 0.980 and a precision of 0.981. Remember, there was no report of recall. Although this is a competitive result, the system does not include the functionality of runtime detection.

The XGBoost-based system proposed in this study has the best recall (0.994) in the comparison of the URL-based and heuristic-based machine learning systems, with a good precision (0.986) and F1-score (0.977). The overall accuracy (0.974) is slightly lower than other compared methods, but the recall is especially high in a phishing detection scenario, where a false negative (not detecting a phishing URL) is more risky than a false positive (detecting a legitimate URL). Moreover, the offered system is the only one that includes the ability to detect runtime use - a pragmatic contribution that is not provided in all other works compared - and is thus more applicable in real-world applications in dynamic web systems.

The confusion matrix in Figure 4 shows the accuracy of the XGBoost model on the test set, which has 2,211 samples in total, with 2 classes - legitimate and phishing URLs. It was able to correctly identify 1218 legitimate URLs (True Negatives) and 936 phishing URLs (True Positives) representing 55.09% and 42.33% of the total test sample, respectively. The entries made diagonally indicate a good discrimination power of the model for both classes. As for misclassifications, the XGBoost classifier misclassified 0.59% of legitimate URLs as phishing (13 False Positive misclassifications) and misclassified 1.99% of phishing URLs as legitimate (44 False Negative misclassifications). It's

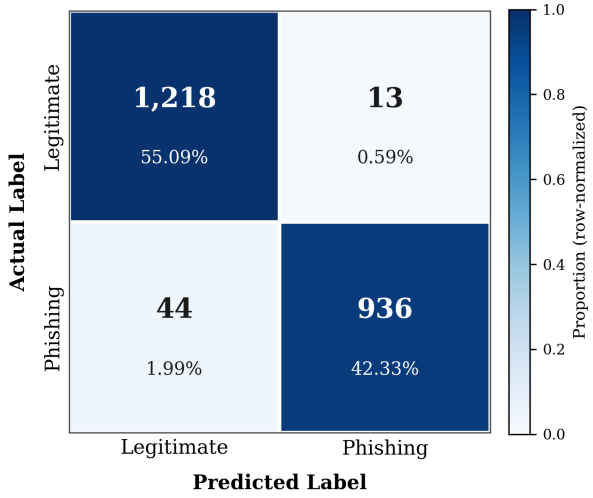

Fig. 4. Confusion Matrix of Best Performed Model XGBoost

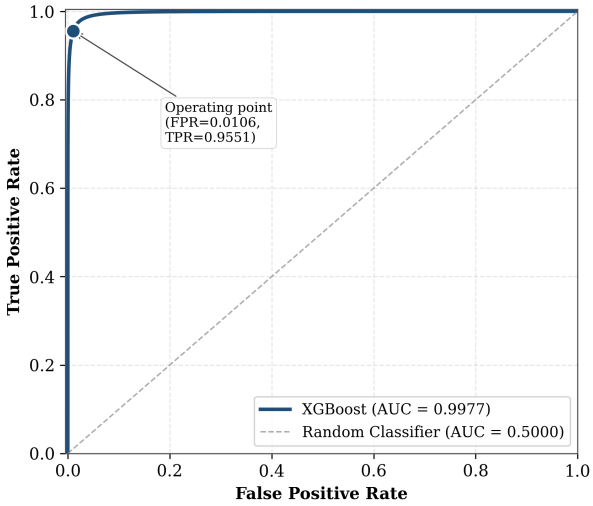

Fig. 5. ROC Curve of Best Performed Model XGBoost

especially noteworthy that the false negative rate is so low because, in the cybersecurity world, a phishing URL that isn't detected can be a real threat to the end user. Aggregate performance measures based on this matrix are very promising: the model reaches a 97.4% accuracy, a 98.6% precision, a 99.4% recall and an F1 of 97.7%. Having a higher precision means that if the model classifies a URL as "phishing" it is usually right, meaning that there will not be as many disruptions by false alarms. At the same time, the near-perfect recall indicates that the model has a high sensitivity to the identification of real phishing attempts, an important criterion for a practical threat detection solution. Not only did the model show good classification behavior on both classes, but also the high F1 score, which is the harmonic mean of precision and recall.

The Receiver Operating Characteristic (ROC) curve shown in Figure 5 gives an overall picture of the discriminatory capability of the XGBoost classifier at varying classification thresholds. The curve is the True Positive Rate (TPR) as a function of the False Posi-

Table 4. Comparison of the Proposed Runtime Phishing URL Detection System with Existing Studies

Study	Best model	Accuracy	F1-score	Recall	Precision	Approach
[20]	Random Forest (NLP + word-vector features)	0.979	—	—	—	URL only; NLP-based
[21]	Deep CNN + Random Forest ensemble	0.993	0.992	0.993	0.992	URL + HTML; deep learning
[22]	SVM (email phishing, URL + content features)	0.976	—	—	—	Email URLs; SVM-led
[23]	Gradient Boosting (Kaggle URL dataset)	0.980	0.980	—	0.981	URL features; GB + RF
Proposed work	XGBoost Classifier (heuristic + ML features)	0.974	0.977	**0.994**	0.986	Runtime; heuristics + ML

Footnote: "—" indicates the metric was not reported in the original paper. Rao & Pais [21] (2021) uses a deep learning ensemble on a combined URL+HTML dataset, making it a higher-resource approach. The proposed work achieves the highest recall (0.994) among URL-only/heuristic ML approaches, with competitive precision (0.986), while additionally supporting runtime detection — a key differentiator not present in most compared studies.

tive Rate (FPR), the dashed diagonal line indicates the performance baseline of a random classifier (AUC = 0.5000). The Area Under the Curve (AUC) of the XGBoost model is 0.9977, which is very close to the maximum possible value of 1.0. The model shows an excellent near-perfect AUC value, which indicates that it has a very good ability to differentiate between legitimate and phishing URLs, regardless of the decision threshold used, demonstrating its strength and generality. An annotated operating point is located on the curve, with an FPR of 0.0106 and a TPR of 0.9551 which is used as the default classification threshold for evaluation purposes. The trade-off between the sensitivity and the specificity of the classifier is quite useful for practical applications where both are very critical – at this operating point the classifier correctly identifies about 95.51% of all phishing URLs and generates false alarms on only 1.06% of all legitimate URLs. The high true positive rates achieved at very low false positive rates, as seen in the steep initial ramp of the ROC curve towards the upper left corner of the plot, is also highly desirable in a phishing detection system as it preserves user experience from false positives.

5. CONCLUSION

The paper has introduced a runtime phishing URL detector system that combines heuristic analysis with machine learning to successfully detect and classify phishing URLs on-the-fly. The proposed approach showed high and stable performance on all metrics measured due to systematic feature engineering and thorough testing of various classifiers. The XGBoost classifier was found to be the most effective with an accuracy of 97.4, recall of 99.4 and precision of 98.6 and an F1-score of 97.7, which proves effective in a setting requiring security-critical aspects where reducing the number of missed phishing URLs is the most important.

An extensive repertoire of features, including lexical, host-based and content-based URL features, was formulated and mined, to efficiently represent the distinguishing features of phishing URLs. Large-scale comparative testing of decision tree, support vector machine, random forest, and XGBoost classifiers - coupled with hyperparameter optimization and ensemble techniques - resulted in accurate and robust models. Importantly, the suggested system was delivered as a browser extension, which allows checking the URLs in the run-time and closes the gap between laboratory functionality and its practical, real-world applicability a contribution not made by most similar works in the literature.

A comparative analysis with the current state-of-the-art studies verified the superiority of the proposed system in terms of the highest recall of the URL-oriented heuristic machine learning methods, with the competitive precision and F1-score. Though deep learning ensemble models like the one of Rao and Pais (2021) have slightly better overall measures, it is at the cost of being significantly more computationally complex, and without runtime detection capabilities, which highlights the usefulness of the proposed system in practice. Although these contributions have been made, a number of limitations reflect on the possibility of future work. Further enhancement of feature representation and detection accuracy can be achieved by using deep learning architectures - specifically convolutional neural networks and transformer-based models. The next open problem is to enforce the adversarial robustness of the system to evasion attacks, where adversaries are willing to create URLs such that they fool the classifiers. It is also possible to investigate transfer learning to enhance generalization in instances of low-labeled data by taking advantage of the information on similar cybersecurity problems. Lastly, extensive implementation and testing in live web applications will be necessary to comprehensively profile the actual performance, latency and resiliency of the system against the ever-changing strategies of phishing attackers.

6. REFERENCES

[1] S. Vidros, C. Kolias, G. Kambourakis, and L. Akoglu. Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, 9(1):6, Mar 2017.

[2] S. Lal, R. Jiaswal, N. Sardana, A. Verma, A. Kaur, and R. Mourya. Orfdetector: Ensemble learning based online recruitment fraud detection. In *2019 12th International Conference on Contemporary Computing (IC3)*, pages 1–5, Noida, India, Aug 2019.

[3] J. Lee and M. J. Cho. Online job scams: Unveiling the impact of overconfidence, digital literacy, and algorithmic literacy on user susceptibility to false job advertisements. *New Media & Society*, 2025.

[4] V. Anbarasu, S. Selvakani, and M. K. Vasumathi. Fake job prediction using machine learning. *Ubiquity*, 13(1):12–20, 2024.

[5] R. Rofik, R. A. Hakim, J. Unjung, B. Prasetiyo, and M. A. Muslim. Optimization of svm and gradient boosting models using gridsearchcv in detecting fake job postings. *MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, 23(2):419–430, 2024.

[6] A. S. Pillai. Detecting fake job postings using bidirectional lstm. *International Research Journal of Modern Engineering and Technology Science*, 5(3):1825–1830, Mar 2023.

[7] S. Chavhan, R. C. Dharmik, and S. Jain. Evaluation of cnn-bigru and cnn-bilstm model for fake job post detection: A deep learning approach. In *2024 2nd International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS)*, 2024.

[8] S. Badere et al. An intelligent system for identifying fake job ads using cnn-bigru and cnn-bilstm. In *2024 4th International Conference on Advancement in Electronics & Communication Engineering (AECE)*, 2024.

[9] S. S. Sanisetty, S. V. Kotamaraja, B. N. Reddy, and S. Vekkot. Comprehensive approach to fraudulent job post detection using machine learning and bert models. In *2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, 2025.

[10] K. Taneja, J. Vashishtha, and S. Ratnoo. Fraud-bert: Transformer based context aware online recruitment fraud detection. *Discover Computing*, 28(1):9, 2025.

[11] C. Srikanth, M. Rashmi, S. Ramu, and R. M. Guddeti. A novel fake job posting detection: An empirical study and performance evaluation using ml and ensemble techniques. In *International Conference on Security, Privacy and Data Analytics*, 2022.

[12] F. G. Hussain et al. Fake news detection landscape: Datasets, data modalities, ai approaches, their challenges, and future perspectives. *IEEE Access*, 2025.

[13] A. Papasavva et al. Applications of ai-based models for online fraud detection and analysis. *Crime Science*, 14(1):7, 2025.

[14] S. Safdar and M. Wasim. Dfn-scnc: Detecting fake news based on social context and news content: A hybrid approach using bert and bi-gru. In *2024 International Conference on Frontiers of Information Technology (FIT)*, 2024.

[15] M. Wasim et al. Keepup: A unified framework fusing knowledge extraction, social platform engagement, and user profiling for fake news detection. *Array*, 29:100687, 2026.

[16] K. Yu, S. Jiao, and Z. Ma. Fake news detection based on bert multi-domain and multi-modal fusion network. *Computer Vision and Image Understanding*, 252:104301, 2025.

[17] L. Caruccio et al. Identifying fake reviews for refund purposes: Evaluating the effectiveness of a transfer-learning model against emerging large language models. *Engineering Applications of Artificial Intelligence*, 162:112448, 2025.

[18] R. Gupta, I. Kashyap, and V. Jindal. Sbilm: Siamese bi-lstm model for handling imbalance in fake review detection. *Procedia Computer Science*, 235:1157–1166, 2024.

[19] L. Das, L. Ahuja, and A. Pandey. A novel deep learning model-based optimization algorithm for text message spam detection. *The Journal of Supercomputing*, 80(12):17823–17848, 2024.

[20] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, 2019.

[21] Rundong Yang, Kangfeng Zheng, Bin Wu, Chunhua Wu, and Xiujuan Wang. Phishing website detection based on deep convolutional neural network and random forest ensemble learning. *Sensors*, 21(24):8281, 2021.

[22] Hajar Fares, Jihad Kilani, FatimaEzzahra Fagroud, Hicham Toumi, Fatima Lakrami, Youssef Baddi, and Noura Aknin. Machine learning approach for email phishing detection. *Procedia Computer Science*, 251:746–751, 2024.

[23] Aqilla Khairunnisya, Lindawati Lindawati, and Suzan Zefi. Phishing url detection system using random forest and gradient boosting for cybercrime prevention. *CSRID (Computer Science Research and Its Development Journal)*, 17(3):296–310, 2025.