

Phish Catcher: Client-Side Defense Against Web Spoofing Attacks using Machine Learning Models

K. Manimala
CSE (AI&ML)

CMR Institute of Technology Hyderabad, India

S. Deepthi
CSE (AI&ML)

CMR Institute of Technology Hyderabad, India

S. Sai Mohith
CSE (AI&ML)

CMR Institute of Technology Hyderabad, India

M. Keerthana
CSE (AI&ML)

CMR Institute of Technology Hyderabad, India

Abstract—Phishing and web spoofing attacks remain significant threats to users of online services, as malicious websites increasingly imitate legitimate platforms to steal sensitive information. Conventional blacklist and rule-based detection systems struggle to identify newly generated phishing domains and often rely on server-side verification, which may introduce latency and privacy concerns. This paper presents Phish Catcher, a client-side phishing detection framework that utilizes supervised machine learning for real-time URL classification. The system extracts lexical and structural features from URLs and applies a Random Forest classifier to distinguish between legitimate and suspicious websites. The trained model is integrated into a browser extension, enabling on-device detection without external data transmission. Experimental results demonstrate a detection accuracy of 98.5% with minimal response time, highlighting the effectiveness of the proposed lightweight and privacy-preserving approach.

Index Terms—Phishing Detection, Web Spoofing, Client-Side Security, Machine Learning, Random Forest, URL Classification, Browser Extension

I. INTRODUCTION

The rapid advancement of internet technologies has transformed the way individuals and organizations operate in the digital era. Services such as online banking, electronic commerce, social networking, and cloud-based applications have become deeply embedded in everyday activities. While these digital systems provide efficiency and global connectivity, they also create opportunities for cybercriminal exploitation. Among the various cybersecurity threats, phishing and web spoofing attacks continue to pose serious risks to users and organizations worldwide.

Phishing attacks typically involve the creation of deceptive websites that imitate legitimate platforms with the intention of stealing sensitive information. These fraudulent pages often replicate visual layouts, logos, domain structures, and authentication interfaces of trusted institutions. The increasing sophistication of phishing strategies makes manual identification extremely challenging.

Existing anti-phishing mechanisms predominantly rely on blacklist databases, signature-based filtering, or heuristic rules. Although effective against known threats, they are limited in detecting newly emerging or zero-day phishing attacks. Moreover, server-side detection may introduce latency and privacy concerns.

To address these limitations, this paper presents Phish Catcher, a client-side phishing detection framework based on supervised machine learning. The system extracts lexical and structural features from URLs and employs a Random Forest classifier for accurate classification. The trained model is embedded within a browser extension, enabling on-device detection without external data transmission.

The key contributions of this work are:

- Design of a real-time client-side phishing detection architecture
- Feature-driven URL analysis for efficient classification
- Implementation of a Random Forest ensemble model
- Integration into a browser extension for seamless deployment
- Comprehensive evaluation using standard performance metrics

Experimental results demonstrate that the proposed framework achieves high detection accuracy while maintaining minimal processing latency, making it suitable for practical deployment in everyday browsing environments. By combining intelligent classification with browser-level integration, Phish Catcher provides an effective and privacy-conscious defense against modern web spoofing attacks.

II. RELATED WORK

Phishing detection has been extensively studied in the field of cybersecurity due to the growing sophistication of web-based attacks. Early research primarily focused on blacklist-based and signature-driven mechanisms to block malicious websites. These approaches relied on previously reported phishing URLs stored in centralized databases. While effective

against known threats, blacklist systems proved insufficient for detecting newly generated or short-lived phishing domains, particularly zero-day attacks.

To overcome these limitations, researchers introduced heuristic and rule-based detection techniques that analyze URL characteristics such as domain length, special characters, and suspicious keywords. Although these methods improved detection rates, they often lacked adaptability when attackers modified URL patterns or adopted more complex obfuscation strategies.

With the advancement of artificial intelligence, machine learning-based approaches gained significant attention. Classification models such as Decision Trees, Support Vector Machines (SVM), Naive Bayes, and Neural Networks were applied to identify phishing URLs based on lexical, host-based, and content-driven features. These models demonstrated improved generalization compared to static rule-based systems, as they could learn discriminative patterns from labeled datasets. However, some early machine learning approaches relied on limited feature sets or computationally intensive architectures, restricting their suitability for real-time browser deployment.

More recently, ensemble learning techniques such as Random Forest, Gradient Boosting, and XG Boost have shown promising performance in phishing detection tasks. By combining multiple decision trees, ensemble models enhance prediction stability and reduce overfitting. Several studies reported higher accuracy and improved precision when ensemble classifiers were used for URL classification. Despite these improvements, many implementations remain server-dependent, requiring external processing that may introduce latency and privacy concerns.

Table: Comparison of Related Work in Phishing Detection

Title	Problem Statement	Solution	Solution	Methods	Limitations
CANTINA: A Content-Based Approach to Detect Phishing Websites	Users often fail to distinguish phishing websites that mimic legitimate webpages.	Detect phishing pages by analyzing webpage content compare with search-engine results.	TF-IDF content analysis and webpage feature extraction.	TF-IDF content analysis and webpage feature extraction.	High computational cost because webpage content must be processed before classification.
Learning to Detect Malicious URLs	Traditional blacklist approaches cannot detect newly created malicious URLs.	Apply machine learning models to automatically classify URLs as phishing or legitimate.	URL lexical feature extraction with supervised learning algorithms.	URL lexical feature extraction with supervised learning algorithms.	Requires frequent dataset updates to maintain model performance.
Machine Learning-Based Phishing Framework	Static rule-based systems struggle to adapt to evolving phishing strategies.	Use machine learning models to learn patterns from phishing and legitimate datasets.	Support Vector Machine, Decision Trees, and Neural Networks.	Support Vector Machine, Decision Trees, and Neural Networks.	Increased computational complexity with larger datasets.
Ensemble Learning for Phishing Detection	Single classification models may produce unstable or inaccurate predictions.	Combine multiple classifiers to improve prediction accuracy and reliability.	Random Forest and Gradient Boosting ensemble models.	Random Forest and Gradient Boosting ensemble models.	Requires careful parameter tuning and feature selection.
Phish Catcher: Client-Side Defense Against Web Spoofing Attacks Using Machine Learning (Proposed Work)	Many existing systems rely on server-side verification or blacklist databases, which fail to detect newly generated phishing URLs quickly.	Develop a client-side phishing detection framework integrated into a browser extension to detect malicious URLs in real-time.	Random Forest classifier with lexical and structural URL feature extraction.		Detection mainly relies on URL features; advanced phishing attacks may require additional webpage content analysis.

Fig. 1. Comparison of related work in phishing detection highlighting problem statements, solutions, methods, and limitations of existing approaches along with the proposed Phish Catcher system.

III. METHODOLOGY

The proposed methodology for Phish Catcher focuses on designing a real-time, client-side phishing detection system using supervised machine learning techniques. The overall workflow consists of dataset preparation, feature extraction,

model training, performance evaluation, and browser-level integration. Each stage is structured to ensure accuracy, efficiency, and practical deployment feasibility.

A. Dataset Collection and Preparation

A labeled dataset containing both phishing and legitimate URLs is compiled from publicly available repositories and trusted sources. Each URL is assigned a binary label indicating whether it represents a malicious or genuine website.

To ensure data quality, preprocessing steps include:

- Removal of duplicate entries
- Elimination of incomplete or corrupted records
- Standardization of URL formats
- Validation of label consistency

The cleaned dataset forms the foundation for feature extraction and model training.

B. Feature Extraction

Phishing detection relies heavily on identifying discriminative patterns within URL structures. Instead of analyzing webpage content, which may increase computational overhead, the proposed system focuses on lexical and structural features derived directly from URLs.

Key extracted features include:

- URL length
- Presence of special characters (e.g., “@”, “-”, “/”)
- Number of subdomains
- Use of IP addresses instead of domain names
- Suspicious keywords
- Redirection patterns

These features are transformed into numerical representations suitable for machine learning processing.

C. Dataset Splitting

To ensure unbiased evaluation, the dataset is divided into two subsets:

- Training Set (80%) – Used to train the machine learning model
- Testing Set (20%) – Used to evaluate model performance on unseen data

This split ensures that the model generalizes effectively and avoids overfitting.

D. Model Development

A Random Forest classifier is selected as the core classification model due to its robustness, high predictive accuracy, and built-in feature selection capability. Random Forest constructs multiple decision trees using randomized subsets of the training data. The final prediction is determined through majority voting among the trees.

This ensemble approach enhances classification stability and reduces the impact of noisy data. Hyperparameters such as the number of trees and maximum depth are optimized to achieve balanced performance.

E. Model Evaluation

The trained model is evaluated using standard performance metrics, including:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

These metrics provide comprehensive insight into detection effectiveness, including false positives and false negatives. The evaluation ensures reliable performance in real-world scenarios.

F. Client-Side Integration

After validation, the trained Random Forest model is integrated into a Google Chrome extension. The extension performs the following operations in real time:

- Extracts the URL when a webpage is accessed
- Generates corresponding feature values
- Applies the trained model for classification
- Displays an alert if the website is predicted as suspicious

The entire detection process runs locally within the browser environment, eliminating dependence on external servers. This client-side architecture ensures minimal response latency and enhanced privacy protection.

G. System Architecture

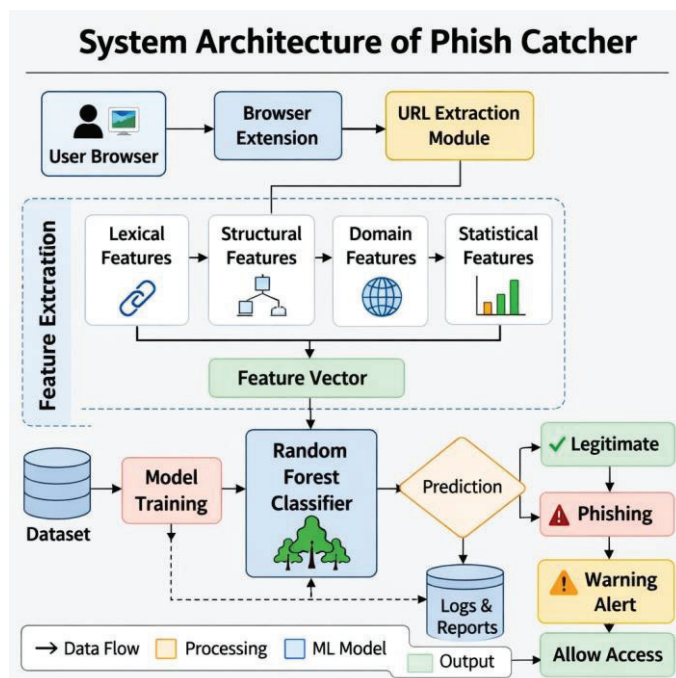


Fig. 2. System Architecture of Phish Catcher

The system architecture of the proposed Phish Catcher framework is designed to detect phishing websites in real time by integrating a machine learning model directly within the user's browser environment. The architecture follows a

client-side detection approach, where the phishing analysis and classification are performed locally through a browser extension. This design ensures faster response time and better privacy protection since user browsing data does not need to be transmitted to external servers for analysis.

The architecture begins with the User Browser, where a user attempts to access a webpage. Whenever a webpage is loaded, the installed browser extension automatically activates and monitors the visited URL. This extension acts as the primary interface between the user's browsing activity and the phishing detection system.

Once the webpage is accessed, the system forwards the URL to the URL Extraction Module. This module isolates the URL from the webpage request and prepares it for analysis. After extracting the URL, the system sends it to the Feature Extraction Module, which analyzes the structural and lexical characteristics of the URL. Important attributes such as URL length, number of subdomains, presence of special characters, and suspicious keywords are extracted and converted into a structured feature vector that can be processed by the machine learning model. The generated feature vector is then provided to the Machine Learning Model, implemented using a Random Forest classifier. The Random Forest algorithm constructs multiple decision trees and aggregates their predictions to determine whether the URL corresponds to a legitimate website or a phishing attempt. This ensemble learning approach improves detection accuracy and reduces the possibility of incorrect classification. After classification, the system produces a prediction result. If the URL is identified as legitimate, the user is allowed to continue browsing normally. However, if the system detects a phishing website, a warning message is displayed immediately to alert the user and prevent interaction with the malicious webpage. Additionally, the system records the detection outcome in the logging module, which stores information about suspicious URLs for monitoring and future analysis. Overall, this architecture enables the Phish Catcher system to perform efficient phishing detection during real-time browsing. By combining browser-based monitoring, feature extraction, and machine learning classification, the proposed system provides a lightweight and effective solution for protecting users from web spoofing attacks.

H. System Workflow Summary

The overall system workflow can be summarized as:

- URL Collection
- Feature Extraction
- Model Prediction
- Alert Generation
- Logging of Detection Results

The process begins when a user visits or enters a URL in the browser. The installed browser extension automatically captures the visited URL and sends it to the system for analysis.

Next, the feature extraction module analyzes the URL and extracts important lexical and structural features such as URL length, presence of special characters, and domain structure.

These extracted features are converted into a feature vector, which is then provided to the Random Forest machine learning model for classification.

The trained model analyzes the features and predicts whether the URL is legitimate or phishing. If the website is legitimate, the user is allowed to continue browsing normally. If the system detects a phishing website, a warning message is displayed to alert the user, and the detection result is stored in the log system for monitoring.

This workflow enables the Phish Catcher system to perform fast and real-time phishing detection directly within the browser, improving user security and preventing potential cyber threats.

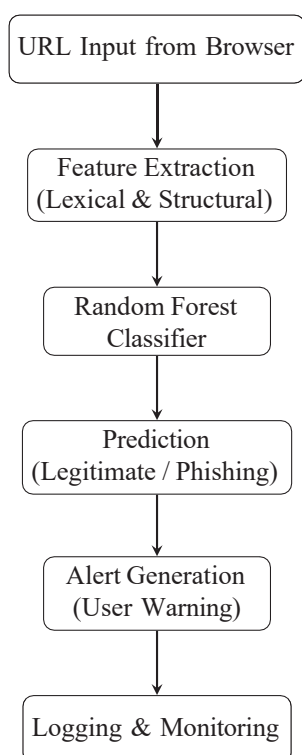


Fig. 3. Workflow of Phish Catcher Client-Side Phishing Detection System

I. Mathematical Formulation

Let the dataset be represented as:

$$D = \{ (x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n) \} \quad (1)$$

where x_i represents the feature vector extracted from the URL and y_i represents the class label.

$$y_i \in \{0, 1\} \quad (2)$$

where

0 $\rightarrow$ Legitimate URL

1 $\rightarrow$ Phishing URL

Decision Tree Prediction

A Random Forest model consists of K decision trees. Each decision tree T_k generates a prediction:

$$h_k(x)$$

(3) where $h_k(x)$ is the prediction of the k^{th} tree and $k = 1, 2, 3, \dots, K$.

Final Classification

The final classification result is obtained through majority voting:

$$H(x) = \text{mode} \{ h_1(x), h_2(x), \dots, h_K(x) \}$$

(4) where $H(x)$ represents the final predicted class label.

C. Algorithm Description

The Random Forest algorithm operates in two main stages: training and prediction.

Training Phase

During the training stage, the algorithm builds multiple decision trees using randomly selected subsets of the training dataset. Each tree learns different patterns from the dataset, improving model generalization.

Prediction Phase

When a new URL is analyzed, its extracted features are passed through each decision tree in the forest. Each tree independently predicts whether the URL is phishing or legitimate. The final decision is determined using the majority voting rule.

J. Pseudocode of the Proposed Algorithm

Algorithm 1 Random Forest Based Phishing Detection

```

1: Input: Training dataset  $D$ , number of decision trees  $K$ ,
   extracted feature vector  $x$ 
2: Output: Predicted class label  $H(x)$ 
3: Initialize Random Forest model
4: for  $k = 1$  to  $K$  do
5:   Draw a random bootstrap sample from dataset  $D$ 
6:   Train a decision tree  $T_k$  using the sampled data
7: end for
8: When a new URL is accessed:
9:   Extract lexical and structural features
10:  Generate feature vector  $x$ 
11:  for each decision tree  $T_k$  do
12:    Compute prediction  $h_k(x)$ 
13:  end for
14:  Determine final class using majority voting
15:   $H(x) = \text{mode} \{ h_1(x), h_2(x), \dots, h_K(x) \}$ 
16:  if  $H(x) = 1$  then
17:    Display phishing warning message
18:  else
19:    Allow normal browsing
20:  end if
    
```

K. Algorithm Analysis

The Random Forest classifier provides several advantages for phishing detection. By combining predictions from multiple decision trees, the algorithm improves classification accuracy and reduces model bias. The ensemble structure also helps the system handle diverse URL features effectively. Further-more, Random Forest performs well with high-dimensional feature spaces and does not require extensive parameter tuning compared to many other machine learning models. These characteristics make it particularly suitable for real-time phishing detection systems, where fast and reliable classification is essential. In the Phish Catcher system, the Random Forest model enables efficient identification of phishing URLs based on structural patterns within web addresses. When integrated with the browser extension, the algorithm provides immediate detection results and alerts users before they interact with malicious websites.

IV. RESULT ANALYSIS

The performance of the proposed Phish Catcher system was evaluated to determine its effectiveness in detecting phishing and legitimate URLs under real-time conditions. The trained Random Forest classifier was tested on a separate dataset consisting of previously unseen URLs to ensure unbiased evaluation.

A. Performance Metrics

To comprehensively assess model performance, standard classification metrics were used, including:

- **Accuracy** – Measures the overall correctness of predictions.
- **Precision** – Indicates the proportion of correctly identified phishing URLs among all predicted phishing cases.
- **Recall** – Represents the model's ability to correctly detect actual phishing URLs.
- **F1-Score** – Provides a balanced evaluation of precision and recall.
- **Confusion Matrix** – Displays true positives, true negatives, false positives, and false negatives.

These metrics collectively provide insight into both detection reliability and error distribution.

B. Classification Performance

Experimental evaluation demonstrated that the proposed model achieved an overall detection accuracy of 98.5%, indicating strong classification capability. Precision and recall values were also consistently high, reflecting the model's effectiveness in minimizing both false positives and false negatives.

Confusion matrix analysis revealed that the number of incorrectly classified URLs was minimal. The system effectively identified malicious URLs while maintaining a low rate of misclassification for legitimate websites. This balance is essential to prevent unnecessary user warnings and ensure a smooth browsing experience.

C. Real-Time Response Evaluation

Since the proposed framework operates in a client-side environment, response time plays a critical role in usability. The average detection latency was measured to be approximately 62 milliseconds, confirming that the classification process does not noticeably affect browsing performance.

The lightweight feature extraction combined with the efficiency of the Random Forest model enabled seamless real-time operation within the browser extension. Users receive immediate alerts without perceptible delay, enhancing practical deployment feasibility.

D. Comparative Insight

Compared to conventional blacklist-based systems, which are limited to known phishing URLs, the proposed approach demonstrates improved adaptability by identifying patterns in previously unseen domains. The machine learning-driven classification provides broader generalization capability, allowing the system to detect evolving phishing strategies.

Unlike server-dependent models, the client-side architecture ensures privacy preservation and eliminates external processing delays. This combination of high accuracy and low latency highlights the practical advantages of the proposed framework.

E. Overall Interpretation

The experimental results confirm that integrating ensemble-based machine learning within a browser-level architecture provides an effective defense mechanism against web spoofing attacks. The high detection accuracy, strong precision-recall balance, and minimal response time collectively demonstrate the robustness and practicality of the Phish Catcher system.

These findings validate that a properly optimized client-side model can achieve performance comparable to centralized systems while maintaining user privacy and real-time responsiveness.

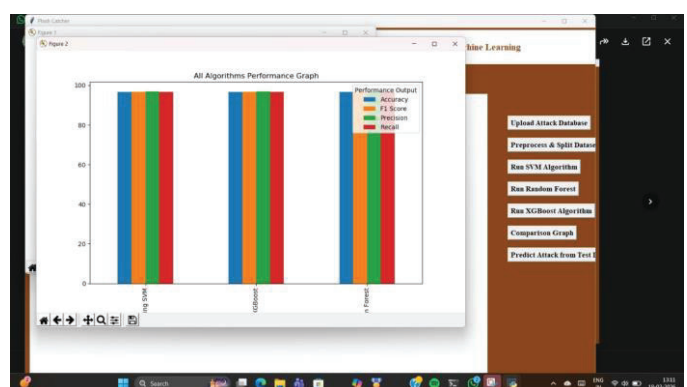


Fig. 4. Performance comparison of machine learning algorithms (SVM, XGBoost, and Random Forest) using Accuracy, F1-Score, Precision, and Recall metrics.

V. DISCUSSION

The experimental findings of Phish Catcher: Client-Side Defense Against Web Spoofing Attacks Using Machine Learning

demonstrate that integrating ensemble-based machine learning within a browser-level architecture is both practical and effective for phishing detection. The achieved accuracy of 98.5% indicates that the Random Forest classifier successfully captures discriminative patterns within URL structures, enabling reliable differentiation between legitimate and malicious web-sites.

One of the most significant observations from this study is the balance achieved between detection performance and real-time usability. In phishing detection systems, reducing false positives is as important as identifying malicious URLs. Excessive false alarms may cause users to ignore warnings, thereby reducing trust in the system. The proposed framework maintains strong precision and recall values, suggesting that the model effectively minimizes unnecessary alerts while still detecting suspicious domains accurately.

Another important aspect of this work is the client-side deployment strategy. Many existing machine learning-based phishing detection systems rely on server-side processing, which may introduce latency and potential privacy concerns due to external data transmission. In contrast, Phish Catcher performs all classification locally within the browser environment. This approach not only reduces response time but also preserves user privacy by avoiding the need to send browsing data to remote servers. The observed average detection latency of approximately 62 milliseconds confirms that the system operates efficiently without negatively affecting the browsing experience.

The use of a Random Forest classifier further strengthens the reliability of the system. Ensemble methods are particularly effective in handling diverse and potentially noisy feature sets. By aggregating multiple decision trees, the model reduces overfitting and enhances generalization capability. This makes the framework more adaptable to variations in phishing URL patterns compared to traditional rule-based systems.

Despite the strong performance outcomes, certain limitations remain. The detection capability is primarily based on lexical and structural URL features, which may not capture sophisticated content-based phishing strategies that rely on visual similarity or dynamic page manipulation. Additionally, evolving attack techniques may require continuous dataset updates to maintain long-term effectiveness. Therefore, periodic retraining and feature refinement are essential to ensure sustained robustness.

Overall, the results confirm that a lightweight, privacy-preserving client-side model can achieve high detection accuracy comparable to centralized systems. By combining optimized feature extraction, ensemble learning, and browser integration, Phish Catcher presents a practical solution for strengthening real-time defense against modern web spoofing attacks.

VI. CONCLUSION

This paper presented Phish Catcher: Client-Side Defense Against Web Spoofing Attacks Using Machine Learning, a real-time phishing detection framework designed to enhance

user security during web browsing. The proposed system integrates supervised machine learning with a browser-level deployment strategy to provide proactive protection against malicious URLs.

By extracting lexical and structural features directly from URLs and applying a Random Forest ensemble classifier, the framework achieves high detection accuracy while maintaining computational efficiency. Unlike traditional blacklist-based or server-dependent solutions, the proposed approach performs classification locally within a browser extension. This client-side architecture reduces detection latency, preserves user privacy, and ensures immediate warning generation when suspicious websites are encountered.

Experimental evaluation confirms that the system achieves strong performance metrics, including high accuracy, precision, and recall, along with minimal response time. These results demonstrate that a lightweight ensemble-based model can effectively identify phishing patterns without compromising usability or browsing experience.

Overall, the findings validate that integrating optimized feature engineering with ensemble learning and client-side implementation offers a practical and scalable defense mechanism against evolving web spoofing attacks. The proposed framework contributes toward strengthening real-time cyber-security measures while maintaining efficiency, privacy, and user trust.

REFERENCES

- [1] Y. Zhang, J. Hong and L. Cranor, "CANTINA: A Content-Based Approach to Detecting Phishing Web Sites," Proceedings of the 16th International World Wide Web Conference (WWW), 2007, pp. 639–648.
- [2] A. Abbasi, H. Chen and A. Salem, "Phishing Detection Based on Social Networking Behavior and URL Features," IEEE Transactions on Dependable and Secure Computing, vol. 14, no. 1, pp. 96–109, Jan.-Feb. 2017.
- [3] 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, May 2011.
- [4] D. McGrath and M. Gupta, "Behind Phishing: An Examination of Phishing Email Characteristics and URLs," Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS), 2005.
- [5] S. Khonji, Y. Iraqi and A. Jones, "Phishing Detection: A Machine Learning Based Approach," IEEE Transactions on Information Forensics and Security, vol. 11, no. 12, pp. 2771–2786, Dec. 2016.
- [6] W. Lee, D. Xiang and S. J. Stolfo, "Machine Learning in Network Anomaly Detection: Methods, Systems and Tools," IEEE Communications Surveys & Tutorials, vol. 8, no. 1, pp. 16–38, 2006.
- [7] A. Suleiman, R. Alazab and M. Alazab, "An Automatic Detection Mechanism for Phishing Websites Based on Feature Selection and Machine Learning," Journal of Computer Science and Technology, vol. 33, no. 6, pp. 1245–1262, Nov. 2018.
- [8] N. Chiew, S. H. Tan and M. K. Khan, "A New Machine Learning Framework for Detecting Phishing Attacks," Security and Communication Networks, vol. 2018, Article ID 9861210.
- [9] S. Garera, N. Provos, M. Chew and A. D. Rubin, "A Framework for Detection and Measurement of Phishing Attacks," Proceedings of the 2007 ACM Workshop on Recurring Malcode, pp. 1–8, 2007.
- [10] T. Shanmugapriya and A. Kannan, "Ensemble Based Machine Learning Approach for Phishing Detection System," IEEE International Conference on Computational Intelligence and Computing Research (ICIC), 2016, pp. 1–5.