Phishfinder: Phishing Detection Plugin

Sruthy Suresh
Dept. of CSE
FISAT Cochin
Kerala, India

Joyal Devassy
Dept. of CSE
FISAT Cochin
Kerala, India

Nikhil Ravi
Dept. of CSE
FISAT Cochin
Kerala, India

Vijay Bhaskar
Dept. of CSE
FISAT Cochin
Kerala, India

Vishal C Varghese
Dept. of CSE
FISAT Cochin
Kerala, India

***Abstract*—The proliferation of phishing websites poses a significant threat to users' online security. In response, we propose a browser extension designed to de- tect and alert users to potential phishing websites in real-time. Leveraging machine learning techniques, specifically a Random Forest model trained on various features extracted from URLs, the extension analyzes website URLs as users navigate the web. Upon detecting a potential phishing website, the extension automatically displays a warning message, requiring no user inter- vention for enhanced ease of use. Additionally, users can interact with the extension by clicking on its icon, which provides detailed information about the safety level of the current website, along with the extracted features con- tributing to its classification. For safe websites, the extension displays the safety level, offering users con- fidence in their browsing experience. This browser extension empowers users to navigate the web securely, miti- gating the risks associated with phishing attacks through proactive detection and informative alerts.**

I. Introduction

In the realm of cybersecurity, the detection of phishing websites poses a significant challenge due to their deceptive nature and evolving tactics. Phishing, characterized by the impersonation of legitimate entities to acquire sensitive information from unsuspecting users, continues to be a prevalent threat in the digital landscape. Traditional approaches to phishing detection often rely on server-side processing or complex machine learning models, which may not be suitable for real-time detection within web browsers.

This literature review focuses on the development of a novel solution, PhishFinder, a real-time phishing website detection system implemented as a lightweight browser plugin. Inspired by the paper "PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System," this project leverages URL analysis and the Random Forest classifier to identify phishing websites directly within the client's browser environment. By analyzing key attributes of URLs and employing machine learning techniques, PhishFinder aims to provide users with immediate protection against phishing threats without compromising their privacy or introducing network latency.

The integration of JavaScript and browser plugins enables PhishFinder to offer real-time detection capabilities, empowering users to browse the web with confidence while mitigating the risk of falling victim to phishing attacks. This literature review explores the technical implementation, effectiveness, and implications of PhishFinder in enhancing web security and combating phishing in the modern digital landscape.

II. Related Works

A. DIRECTORY BASED APPROACHES

Most popular one of this kind is PhishTank. According to PhishTank , it is a collaborative clearing house for data and information about phishing on the Internet. Also, PhishTank provides an open API for developers and researchers to integrate anti-phishing data into their applications at no charge. Thus PhishTank is a directory of all phishing websites that are found and reported by people across the web so that developers can use their API for detecting phishing websites. Google has a API called Google Safe Browsing API which also follows directory based approach and also provides open API similar to PhishTank. This kind of approach clearly can't be effective as new phishing web sites are continuously developed and the directory can't be kept up to date always. This also leaks users browsing behaviour as the URLs are sent to the PhishTank API.

B. RULE BASED APPROACHES

An existing chrome plugin named PhishDetector uses a rule based approach so that it can detect phishing without external web service. Although rule based approaches support easier implementation on client side, they can't be accurate compared to Machine Learning based approaches. Similar work by Shreeram.V on detection of phishing attacks using genetic algorithm uses a rule that is generated by a genetic algorithm for detection. PhishNet is one such Predictive blacklisting approach. It used rules that can match with TLD, directory structure, IP address, HTTP header response and some other. SpoofGuard by Stanford is a chrome plugin which used similar rule based approach by considering DNS, URL, images and links. Phishwish: A Stateless Phishing Filter Using Minimal Rules by Debra L. Cook, Vijay K. Gurbani, Michael Daniluk worked on a phishing filter that offers advantages over existing filters: It does not need any training and does not consult centralized white or black lists. They used only 11 rules to determine the veracity of an website.

C. ML BASED APPROACHES

Intelligent phishing website detection using random forest classifier (IEEE-2017) by Abdulhamit Subasi, Esraa Molah, Fatin Almkallawi and Touseef J. Chaudhery discusses the use the random forest classifier for phishing detection. Random Forest has performed the best among the classification methods by achieving the highest accuracy 97.36PhishBox:

An Approach for Phishing Validation and Detection (IEEE-2017) by Jhen-Hao Li, and Sheng-De Wang discusses ensemble models for phishing detection. As a result, The false-positive rate of phishing detection is dropped by 43.7Real time detection of phishing websites (IEEE-2016) by Abdulghani Ali Ahmed, and Nurul Amirah Abdullah discusses an approach based on features from only the URL of the website. They were able to come up with a detection mechanism that is capable of detecting various types of phishing attacks maintaining a low rate of false alarms. Using Domain Top-page Similarity Feature in Machine LearningBased Web Phishing Detection by Nuttapong Sanglerdsinlapachai, Arnon Rungsawang presents a study on using a concept feature to detect web phishing problem. They applied additional domain top-page similarity feature to a machine learning based phishing detection system. The evaluation result in terms of f-measure was up to 0.9250, with 7.50% of error rate. Netcraft is one popular phishing detection plugin for chrome that uses server side prediction.

D. DRAWBACKS

Based on the above mentioned related works, It can be seen that the plugins either use rule based approach or server side ML based approach. Rule based approach doesn't seem to perform well compared to ML based approaches and on the other side ML based approaches need libraries support and so they are not implemented in client side plugin. All the existing plugins send the target URL to an external web server for classification. This project aims to implement the same in browser plugin removing the need of external web service and improving user privacy.

III. Proposed System

The block diagram of the entire system is shown in the figure. A Random Forest classifier is trained on phishing sites dataset using python scikit-learn. A JSON format to represent the random forest classifier has been devised and the learned classifier is exported to the same. A browser script has been implemented which uses the exported model JSON to classify the website being loaded in the active browser tab. The system aims at warning the user in the event of phishing. Random Forest classifier on 17 features of a website is used to classify whether the site is phishing or legitimate. The dataset arff file is loaded using python arff library and 17 features are chosen from the existing 30 features. Features are selected on basis that they can be extracted completely offline on the client side without being dependent on a web service or third party. The dataset with chosen features are then separated for training and testing. Then the Random Forest is trained on the training data and exported to the above mentioned JSON format. The JSON file is hosted on a URL. The client side chrome plugin is made to execute a script on each page load and it starts to extract and encode the above selected features. Once the features are encoded, the plugin then checks for the exported model JSON in cache and downloads it again incase it is not there in cache.

A. UI DESIGN

A simple and easy to use User Interface has been designed for the plugin using HTML and CSS. The UI contains a large circle indicating the percentage of the legitimacy of the website in active tab. The circle also changes its colour with respect to the classification output (Green for legitimate website and Light Red for phishing). Below the circle, the analysis results containing the extracted features are displayed in the following colour code. Green - Legitimate Yellow - Suspicious Light Red - Phishing The plugin also displays a alert warning incase of phishing to prevent the user from entering any sensitive information on the website. The test results such as precision, recall and accuracy are displayed in a separate screen. The UI is shown in figure

B. CLASS DIAGRAM

The class diagram of the entire Machine Translation system is shown in figure. This diagram depicts the functions of various modules in the system clearly. It also shows the interaction between the modules of the system thereby providing a clear idea for implementation.

C. MODULE DESIGN

1) Preprocessing: The dataset is downloaded from UCI repository and loaded into a numpy array. The dataset consists of 30 features, which needs to be reduced so that they can be extracted on the browser. Each feature is experimented on the browser so that it will be feasible to extract it without using any external web service or third party. Based on the experiments, 17 features have been chosen out of 30 without much loss in the accuracy on the test data. More number of features increases the accuracy and on the other hand, reduces the ability to detect rapidly considering the feature extraction time. Thus a subset of features is chosen in a way that the tradeoff is balanced.

2) Training: The training data from the preprocessing module is loaded from the disk. A random forest classifier is trained on the data using scikitlearn library. Random Forest is an ensemble learning technique and thus an ensemble of 10 decision tree estimators is used. Each decision tree follows CART algorithm and tries to reduce the gini impurity. The cross validation score is also calculated on the training data. The F1 score is calculated on the testing data. Then the trained model is exported to JSON using the next module.

3) Exporting Model: Every machine learning algorithm learns its parameter values during the training phase. In Random Forest, each decision tree is an independent learner and each decision tree learns node threshold values and the leaf nodes learn class probabilities. Thus a format needs to be devised to represent the Random Forest in JSON. The overall JSON structure consists of keys such as number of estimators, number of classes and etc. Further it contains an array in which each value is an estimator represented in JSON. Each decision tree is encoded as a JSON tree with nested objects containing threshold for that node and left and right node objects recursively.

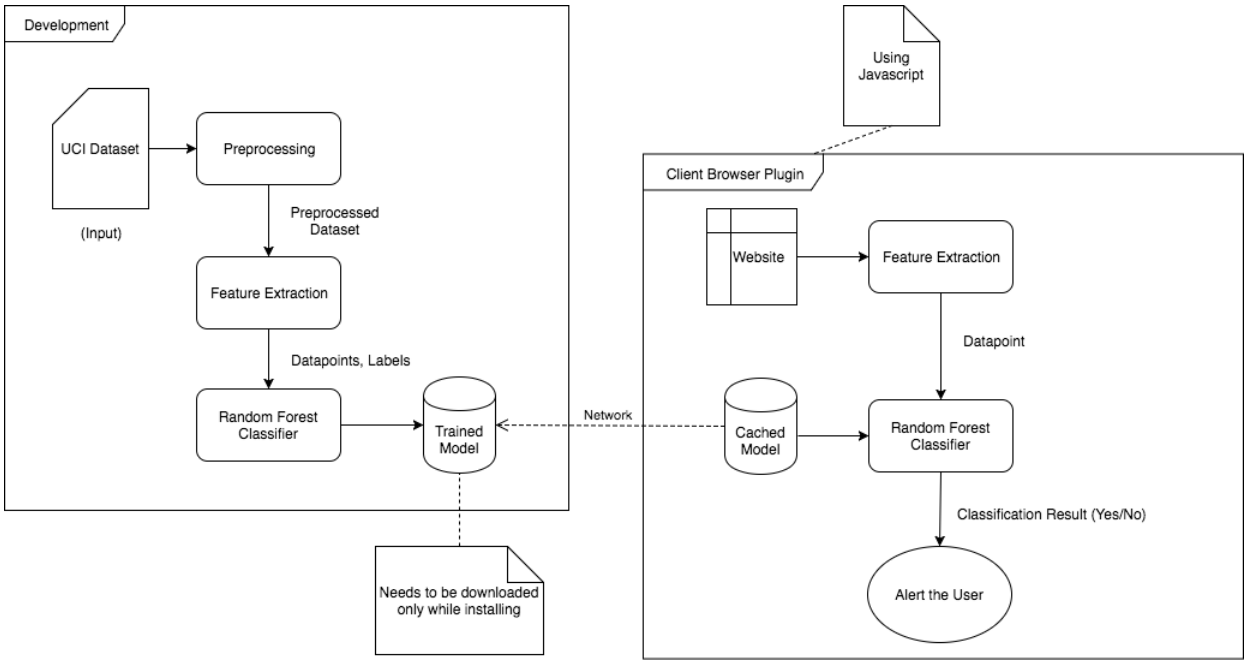


Fig. 1. System Architecture of the proposed model.

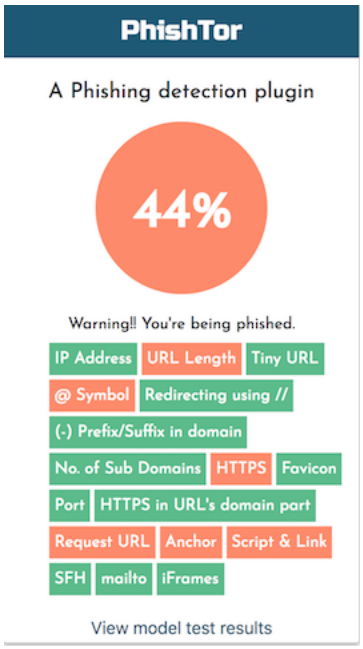


Fig. 2. UI Design

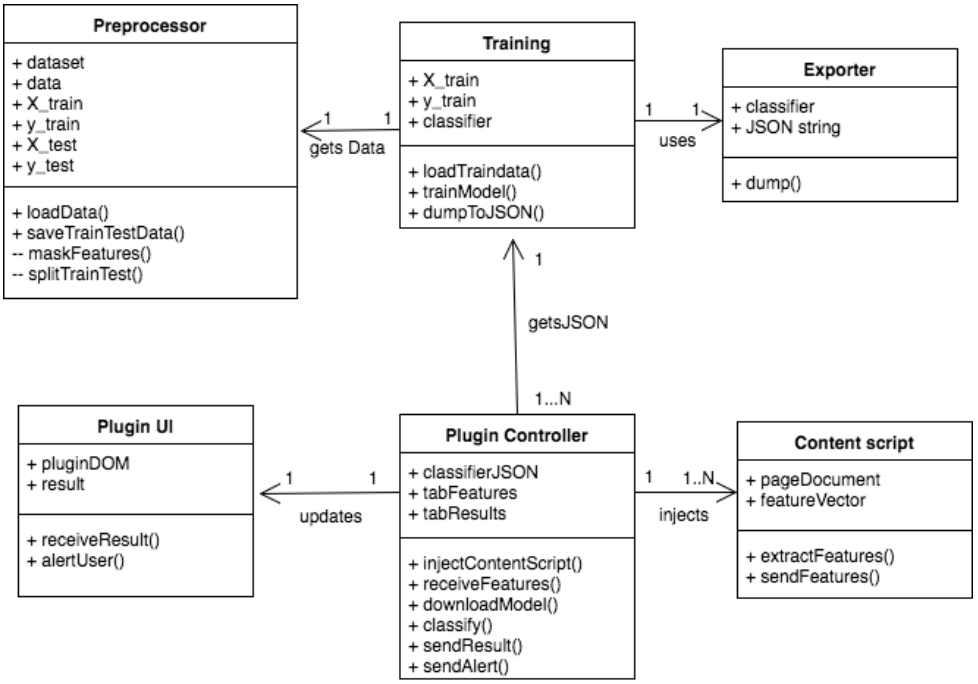


Fig. 3. Class Diagram

4) Plugin Feature Extraction: The above mentioned 17 features needs to be extracted and encoded for each webpage in realtime while the page is being loaded. A content script is used so that it can access the DOM of the webpage. The content script is automatically injected into each page while it loads. The content script is responsible to collect the features and then send them to the plugin. The main objective of this work is not to use any external web service and the features needs to be independent of network latency and the extraction should be rapid. All these are made sure while developing techniques for extraction of features. Once a feature is extracted it is encoded into values -1, 0, 1 based on the following notation. -1 - Legitimate 0 - Suspicious 1 - Phishing The feature vector containing 17 encoded values is passed on to the plugin from the content script.

5) Classification: The feature vector obtained from the content script is ran through the Random Forest for classification. The Random Forest parameters JSON is downloaded and cached in disk. The script tries to load the JSON from disk and incase of cache miss, the JSON is downloaded again. A javascript library has been developed to mimic the Random Forest behaviour using the JSON by comparing feature vector against the threshold of the nodes. The output of the binary classification is based on the leaf node values and the user is warned if the webpage is classified as phishing.

IV. RESULTS AND DISCUSSION

A. Training:

The model obtained a accuracy of .94 and a Cross validation score of .94 in training phase.

B. Plugin Feature Extraction :

The process described involves extracting features from a webpage and structuring them as key-value pairs, with values encoded within the range of -1 to 1. Features represent specific attributes or characteristics of the webpage deemed relevant for analysis, such as page length, keyword presence, or image count. Each feature is associated with a unique identifier (key) and its corresponding value. Encoding values within the range of -1 to 1, known as normalization, ensures uniformity and comparability across different features with varying scales. This approach facilitates efficient data organization and analysis, enabling researchers or algorithms to systematically evaluate and interpret webpage attributes for a variety of purposes, from search engine optimization to content analysis..

C. Classification :

In the Plugin UI, the output of the classification is visually represented by distinct symbols and colors to convey the legitimacy of a website. A green circle typically indicates that the website is deemed legitimate, implying that it poses no immediate threat to users. This designation offers users reassurance that they can safely interact with the site without concerns about potential phishing attempts or malicious activity. Conversely, a light red circle serves as a warning sign, signaling that the website has been flagged as potentially malicious or indicative of phishing activity. This color coding system provides users with clear and intuitive feedback regarding the safety status of websites they encounter, empowering them to make informed decisions about their online interactions and safeguarding against potential cybersecurity threats.

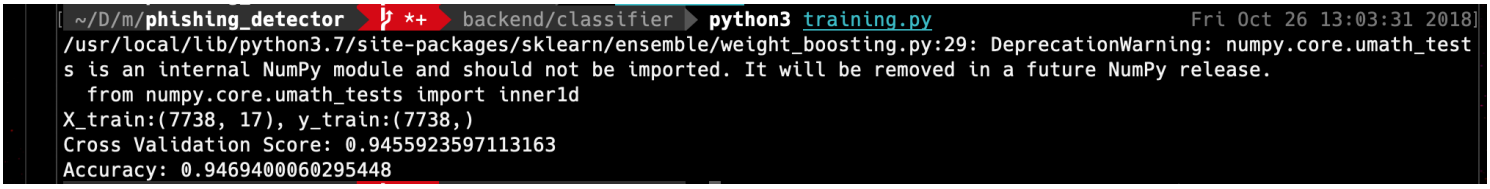
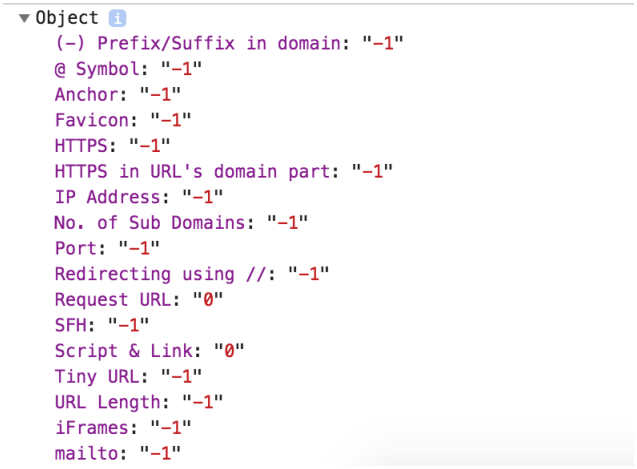


Fig. 4. Class Diagram



V. Conclusion and Future Works

The classifier is currently trained on 17 features which can be increased provided that, they don't make the detection slower or result in loss of privacy. The extension can made to cache results of frequently visited sites and hence reducing computation. But this may result in pharming attack being undetected. A solution needs to be devised for caching of results without losing the ability to detect pharming. The classification in javascript can be done using WorkerThreads which may result in better classification time. Thus a lot of improvements and enhancements are possible this system offers a more usable solution in the field of phishing detection. The PhishFinder project is a groundbreaking initiative in online security, uti- lizing ad- vanced machine learning techniques for swift and efficient detection of phishing attempts. Its primary goal is to deliver a precise and user-friendly system that addresses the dy- namic nature of the digital landscape. The project emphasizes speed and accuracy in phishing threat detection, em- ploying a dual-method approach with immediate Phishtank database checks and comprehensive feature extraction and model-based analysis. The system prioritizes user convenience with an intuitive interface, recognizing the need for ease of use in navigating online platforms. PhishFinder contributes significantly to cybersecurity by safeguarding individuals and organizations from the growing sophistica- tion of phishing attacks. The project's itera- tive development, informed by user feedback, highlights its adaptability and commitment to continuous improvement. Privacy and transparency mea- sures have been implemented to ensure user data secu- rity, enhancing the system's overall reliability and trustworthiness. PhishFinder not only repre- sents a technological advancement but also serves as a proactive response to the es- calating challenges posed by cyber threats. By providing a real-time defense mechanism against phishing, the project marks a significant stride toward fostering a more secure and resilient online environment globally.

VI. References

1) S. Ariyadasa, S. Fernando and S. Fernando, "Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML," in IEEE Access, vol. 10, pp. 82355-82375, 2022, doi: 10.1109/ACCESS.2022.3196018. genes
2) J. Feng, L. Zou, O. Ye and J. Han, "Web2Vec: Phishing Webpage Detection Method Based on Multidimensional Features Driven by Deep Learning," in IEEE Access, vol. 8, pp. 221214-221224, 2020, doi: 10.1109/ACCESS.2020.3043188.
3) L. R. Kalabarige, R. S. Rao, A. R. Pais and L. A. Gabralla, "A Boosting-Based Hybrid Feature Selection and Multi-Layer Stacked Ensemble Learning Model to Detect Phishing Websites," in IEEE Access, vol. 11, pp. 71180-71193, 2023, doi: 10.1109/ACCESS.2023.3293649.
4) Y. Fang, C. Zhang, C. Huang, L. Liu and Y. Yang, "Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism," in IEEE Access, vol. 7, pp. 56329-56340, 2019, doi: 10.1109/ACCESS.2019.2913705.
5) W. Ali and S. Malebary, "Particle Swarm Optimization-Based Feature Weighting for Improving Intelligent Phishing Website Detection," in IEEE Access, vol. 8, pp. 116766-116780, 2020, doi: 10.1109/ACCESS.2020.3003569.
6) E. S. Gualberto, R. T. De Sousa, T. P. De Brito Vieira, J. P. C. L. Da Costa and C. G. Duque, "The Answer is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering," in IEEE Access, vol. 8, pp. 223529-223547, 2020, doi: 10.1109/ACCESS.2020.3043396.
7) M. Sánchez-Paniagua, E. F. Fernández, E. Alegre, W. Al-Nabki and V. González-Castro, "Phishing URL Detection: A Real-Case Scenario Through Login URLs," in IEEE Access, vol. 10, pp. 42949-42960, 2022, doi: 10.1109/ACCESS.2022.3168681.
8) S. Al-Ahmadi, A. Alotaibi and O. Alsaleh, "PDGAN: Phishing Detection With Generative Adversarial Networks," in IEEE Access, vol. 10, pp. 42459-42468, 2022, doi: 10.1109/ACCESS.2022.3168235.
9) Liu and J. Fu, "SPWalk: Similar Property Oriented Feature Learning for Phishing Detection," in IEEE Access, vol. 8, pp. 87031-87045, 2020, doi: 10.1109/ACCESS.2020.2992381.
10) item M. Sameen, K. Han and S. O. Hwang, "PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System," in IEEE Access, vol. 8, pp. 83425-83443, 2020, doi: 10.1109/ACCESS.2020.2991403.