PHISHING WEBSITE DETECTION USING MACHINE LEARNING

Prof. Pooja P[1], Harshith Reddy G[2], Hrithik K[3], Devanapalli Nikhil[4], Bala Sai Yaswanth Yadav[5]

[1] *Assistant Professor, Dept of CS&E, Bangalore Institute of Technology, Karnataka, India*
[2] *Student, Dept of CS&E, Bangalore Institute of Technology, Karnataka, India*
[3] *Student, Dept of CS&E, Bangalore Institute of Technology, Karnataka, India*
[4] *Student, Dept of CS&E, Bangalore Institute of Technology, Karnataka, India*
[5] *Student, Dept of CS&E, Bangalore Institute of Technology, Karnataka, India*

ABSTRACT

With the increasing integration of the Internet into our social and professional lives, we are exposed to a growing number of serious security attacks. One of the primary concerns that needs immediate attention is the ability to detect various network threats, particularly those that involve attacks not seen before. Phishing, in particular, poses a significant risk as attackers aim to collect private information such as user identities, passwords, and financial transactions by creating websites that closely resemble legitimate ones. Government and financial organizations, being popular online destinations for users, have experienced a significant increase in phishing threats and attacks over the years.To combat these evolving phishing methods, it is crucial to employ anti-phishing techniques that can effectively detect and prevent such attacks. Machine learning presents a powerful tool for countering phishing assaults. Various machine learning approaches have been proposed and implemented to identify phishing websites. In this project, three supervised classification models are utilized: Support Vector Machine, Random Forest Classifier, and Logistic Regression.Among these models, the Random Forest Classifier has been found to provide the highest accuracy for the selected dataset, achieving an accuracy score of 97%. This classifier leverages the power of ensemble learning by combining multiple decision trees to make predictions. By using a combination of features extracted from URLs and employing the trained model, it becomes possible to accurately classify websites as phishing or non-phishing.By implementing these machine learning techniques, the project aims to enhance the detection and prevention of phishing attacks, thereby mitigating the risks associated with online security threats.

Keywords: - *Phishing Detection , URLs, Machine Learning, Support Vector Machine, Random Forest, Linear Regression*

1. INTRODUCTION

Phishing is a type of online identity theft that costs internet users billions of dollars each year. Phishers use various techniques to trick unsuspecting internet users into divulging personal information, such as usernames and passwords, by creating fake websites that mimic legitimate ones. To combat this issue, machine learning techniques can be used to detect phishing websites by analyzing their content features, host properties, and page importance properties. By evaluating these features using different machine learning algorithms, we can fine-tune the parameters and select the most effective algorithm for distinguishing between phishing and benign sites. The phishers create unauthorized replicas of real websites and emails, using logos and slogans from legitimate companies, and send them to as many people as possible to lure them into their scheme. When users click on the links or open the websites, they are redirected to a fake website that appears to be legitimate, and their personal information is stolen. Detecting phishing websites is often done through a directory of malicious sites, but machine learning techniques can be more effective. The random forest classifier is a particularly effective machine learning technique for this purpose. Browser plugins can warn users in real-time of potential phishing sites as they browse, but these plugins have restrictions and can compromise user privacy. Existing plugins send URLs to a server for classification, but this can cause delays and may not be as effective. To address these issues, a Web Application

along with a Chrome browser plugin was developed that can classify phishing websites without an external server. This plugin offers improved security and privacy for users.

2. LITERATURE SURVEY

Our literature survey revealed that several common methods are employed in the detection of phishing websites. One prevalent approach involves analyzing the website's URL, where researchers have focused on URL-based features such as domain similarity, subdomain analysis, and presence of suspicious keywords. Another approach utilizes machine learning algorithms, where features like content-based features (e.g., HTML and JavaScript analysis), visual similarity, and lexical analysis have been explored. Furthermore, researchers have also investigated website reputation-based techniques, leveraging blacklists, whitelists, and reputation databases to identify suspicious or known phishing websites. Other notable methods include the use of behavioral analysis, user-centric approaches like browser extensions, and collaborative detection techniques involving crowdsourcing and social media mining. Overall, these diverse methods have contributed to the development of effective phishing website detection systems, each with its strengths and limitations, and continue to evolve as phishing techniques advance.

3. ARCHITECTURE

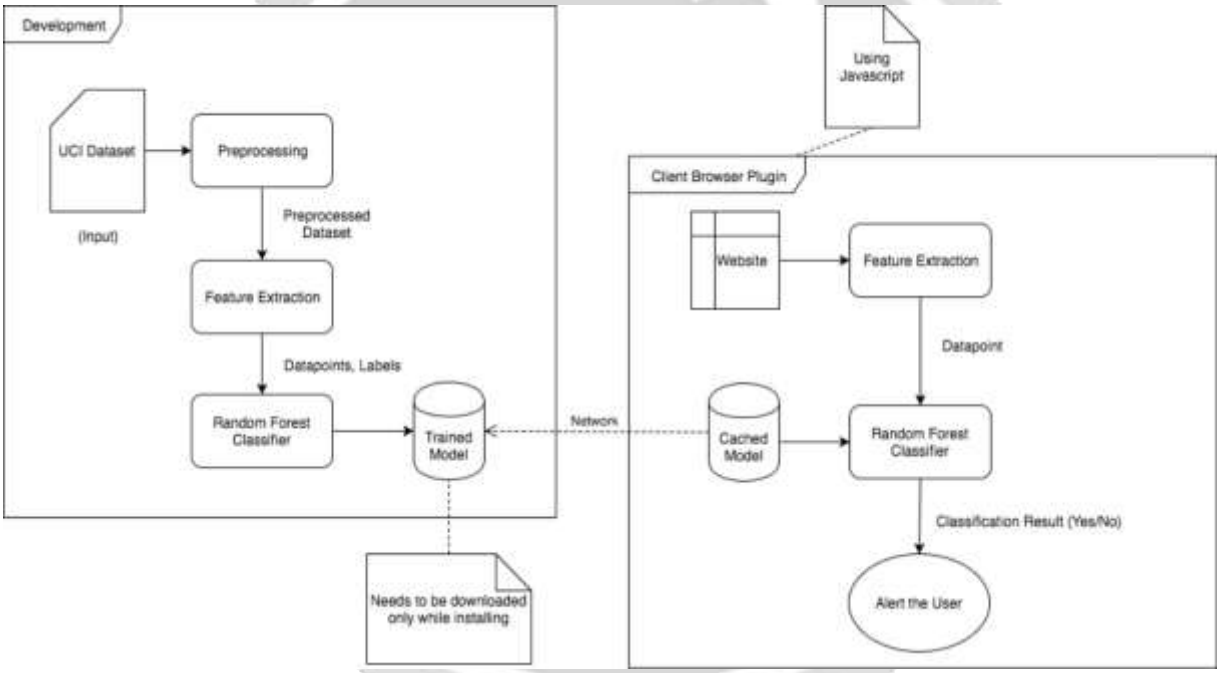

Fig -2: System Architecture

Our Python scikit-learn implementation employs a Random Forest classifier to train on a dataset of phishing websites. To facilitate browser-based phishing detection, a JSON representation of the trained classifier is created and exported. A browser script is developed, utilizing the exported model JSON to classify websites loaded in the active browser tab. The primary objective of this system is to promptly warn users about potential phishing attempts. The Random Forest classifier utilizes 31 features extracted from a website to determine whether it is phishing or legitimate. To ensure offline extraction on the client side without external dependencies, 17 features are selected from the original 30. The dataset in ARFF format is loaded using the Python ARFF library, and the chosen features are separated into training and testing sets. The Random Forest is then trained on the training data and exported to the JSON format mentioned earlier. This JSON file is hosted on a specific URL. To enable seamless integration, a Chrome plugin with minimal resource usage is created. The plugin executes a script every time a webpage is loaded, enabling the extraction and encoding of the chosen features. It efficiently checks the cache for the presence of the exported model JSON and downloads it if required. Utilizing the encoded feature vector and model JSON, the script performs the classification process. If the website is classified as phishing, a warning is promptly displayed to the

user. This system is meticulously designed to prioritize rapid detection and response, ensuring enhanced user protection against phishing attacks.

4. METHODOLOGY

The first module in this project is the splitting module. It involves dividing the dataset into two sets: one containing the dependent variables (the target variable that the model aims to predict) and the other containing the independent variables (the features used for prediction). For example, in the context of determining whether a website is a phishing site or not, the dependent variable could be the phishing label, while the independent variables could include URL length, presence of certain keywords, etc. The dataset is split using the "train_test_split" method, randomly assigning 75% of the data to the training set and 25% to the testing set, ensuring a representative sample in each. The pre-processing module focuses on preparing the data for training. In this project, the chosen dataset has already undergone pre-processing, requiring no further cleaning or missing data handling. However, one step performed during pre-processing is feature scaling, which ensures that the features have a similar scale to prevent any bias or dominance of certain features during training. Feature scaling techniques like normalization or standardization can be applied to achieve this. Once the pre-processed data is ready, it is loaded from disk for training. In this project, three different classifiers are trained on the data using the scikit-learn library. One of the classifiers used is a random forest, which is an ensemble learning technique. Here, an ensemble of 10 decision tree estimators is utilized to improve accuracy and robustness. Using the training data, the models are trained to learn the patterns and relationships between the independent variables and the target variable. The model with the best accuracy score is selected for further use. After the model is trained, it can be exported for future use or evaluation. The exporting module saves the trained model using the pickle module, which allows for efficient object serialization in Python. The model is stored in a file with the ".pkl" extension. This exported model can be later loaded into memory for making predictions on new data. By loading the saved model using the pickle module, the trained model's predict function can be applied to classify new data based on the learned patterns. In addition to the mentioned modules, two more modules can be identified based on the provided context: The feature extraction module is responsible for extracting relevant information from the input data. In this project, various Python modules like whois, requests, socket, re, ipaddress, BeautifulSoup, etc are used to extract features. These modules help gather information such as IP addresses, URL length, domain names, subdomains, presence of favicons, etc. The extracted values are then stored in a list or data structure suitable for further processing. This module plays a crucial role in converting the input URL into a format compatible with the trained classifier, which expects a list of 30 elements representing the respective features. The classification module employs the trained model to predict the target variable based on the provided features. The extracted feature values from the previous module are fed into the trained classifier, which includes algorithms such as Logistic Regression, Support Vector Machines (SVM), and random forest classifier. The classifier uses the learned patterns and relationships from the training phase to classify new data instances as either phishing or non-phishing sites, based on the features provided.
These six modules collectively form the workflow of the project, encompassing data splitting, pre-processing, training, exporting the model, feature extraction, and classification. Each module serves a specific purpose in achieving accurate predictions for identifying phishing websites.

4.1 Random Forest Model

Random Forest is an ensemble learning technique known for its ability to combine multiple decision trees to make predictions. In this project, the focus is on using a random forest classifier to categorize websites as either phishing or non-phishing. The classifier constructs an ensemble of decision trees by randomly selecting subsets of the available data and features for each individual tree. During the prediction phase, each tree in the ensemble independently evaluates the input data, and the final prediction is determined based on the majority vote among the trees. Random Forests offer advantages such as robustness against overfitting, effective handling of high-dimensional data, and the capacity to capture complex relationships between features. In this particular project, two parameters are utilized: the "N_estimators" parameter, which represents the number of decision trees included in the ensemble (in this case, set to 10), and each decision tree is trained independently on a different subset of the training data

4.2 Support Vector Machine

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for classification and regression analysis. In this project, SVM is employed as another classifier to determine phishing websites. SVM constructs a hyperplane in a high-dimensional feature space that optimally separates the different classes. It aims to maximize the

margin between the support vectors (data points closest to the decision boundary) of different classes. SVM is effective in handling high-dimensional data and can handle both linear and non-linear decision boundaries through the use of different kernels. Parameters used: Kernel –this parameter in the algorithm is set a value 'rbf' and therefore non-colinear method is considered.

1	having_IP_Address	16	SFH
2	URL_Length	17	Submitting_to_email
3	Shortining_Service	18	Abnormal_URL
4	having_At_Symbol	19	Redirect
5	double_slash_redirecting	20	on_mouseover
6	Prefix_Suffix	21	RightClick
7	having_Sub_Domain	22	popUpWidnow
8	SSLfinal_State	23	Iframe
9	Domain_registeration_length	24	age_of_domain
10	Favicon	25	DNSRecord
11	port	26	web_traffic
12	HTTPS_token	27	Page_Rank
13	Request_URL	28	Google_Index
14	URL_of_Anchor	29	Links_pointing_to_page
15	Links_in_tags	30	Statistical_report

Fig -1: Features extracted from URLs

4.2 Linear Regression

Logistic Regression is a widely used statistical model for binary classification, effectively capturing the relationship between independent variables and the probability of a specific outcome. In the context of this project, Logistic Regression serves as one of the classifiers to determine if a website is a phishing site. By applying a logistic function, it estimates the probability of the binary outcome, taking into account the input features. Logistic Regression is known for its computational efficiency and interpretability, making it a popular choice for classification tasks. The parameters in this project are set to their default values, ensuring a standard implementation of the algorithm.

5. RESULTS

This section reports the results of our model. The analysis based on parameters such as model accuracy, precision, recall confusion matrix is also presented in this section.

Sl. No	Algorithm	Accuracy
1	Random Forest	0.977
2	Support Vector Machine	0.965
3	Logistic Regression	0.924

Chart -1: Accuracy comparison

Accuracy is a widely adopted metric for evaluating machine learning models, quantifying the ratio of correctly classified instances to the total number of instances. It serves as a measure of the model's effectiveness in predicting the accurate class labels within a classification task. A higher accuracy value indicates a more accurate model. Evaluating your model with only Accuracy cannot be relied upon, especially when dealing with imbalanced

datasets. To obtain a comprehensive evaluation of the model's effectiveness, it is advisable to incorporate additional evaluation metrics like precision, recall, and F1 score alongside accuracy. This ensures a more holistic assessment of the model's performance and its ability to correctly identify both positive and negative instances.

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

$$Precision = \frac{TP}{TP + FP}$$

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

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where, TN – True Negative, TP - True Positive, FP – False Positive, FN – False Negative

Precision: Precision: Precision measures the ratio of correctly identified positive instances among all instances that are predicted as positive. It demonstrates the classifier's capability to minimize false positives, ensuring that the positive predictions are accurate.

Recall: Recall, also referred to as sensitivity or true positive rate, calculates the proportion of correctly identified positive instances out of all the actual positive instances. It showcases the classifier's ability to avoid false negatives, ensuring that positive instances are not missed.

F1 Score: The F1 score is a balanced metric that combines precision and recall into a single value. It is calculated as the harmonic mean of precision and recall, providing an overall assessment of the classifier's performance by considering both aspects. The F1 score is useful when there is an imbalance between positive and negative instances in the dataset, as it takes into account both false positives and false negatives.

	precision	recall	f1-score	support
-1	0.92	0.92	0.92	828
1	0.93	0.93	0.93	920
accuracy			0.92	1748
macro avg	0.92	0.92	0.92	1748
weighted avg	0.92	0.92	0.92	1748

Fig -3: Precision, recall and f1-score for Logistic Regression model

	precision	recall	f1-score	support
-1	0.97	0.96	0.96	828
1	0.96	0.97	0.97	920
accuracy			0.97	1748
macro avg	0.97	0.96	0.96	1748
weighted avg	0.97	0.97	0.97	1748

Fig -4: Precision, recall and f1-score for Support Vector Machine model

	precision	recall	f1-score	support
-1	0.97	0.98	0.98	828
1	0.98	0.98	0.98	920
accuracy			0.98	1748
macro avg	0.98	0.98	0.98	1748
weighted avg	0.98	0.98	0.98	1748

Fig -5: Precision, recall and f1-score for Random Forest model

The above figure shows the precision, recall and f1-scores for the Random Forest model. It is demonstrated that Random Forest Model produced the best accuracy among all the three models.

5. CONCLUSIONS

Phishing has emerged as a significant threat in today's rapidly advancing technological world. As countries shift towards cashless transactions, online businesses, and paperless tickets, phishing poses a major obstacle to this progress. This has led to a decline in people's trust in the reliability of the internet. However, leveraging the power of AI can help gather data and develop effective solutions. For an average person lacking knowledge about security threats, conducting financial transactions online can be risky. Phishers primarily target the payment industry and cloud services, further exacerbating the problem.

In light of these challenges, this project aims to address the issue by demonstrating the use of machine learning in identifying phishing websites. The goal is to create an efficient, accurate, and cost-effective phishing detection mechanism using various machine learning tools and techniques. The project was implemented in the Anaconda IDE and written in Python. Three machine learning classifiers were utilized, and a comparative study of these algorithms was conducted. Notably, the project achieved a high accuracy score, highlighting the effectiveness of the proposed method.

By employing machine learning and conducting thorough research, this project contributes to the ongoing efforts to combat phishing and enhance online security.

6. REFERENCES

[1]. Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, Banu Diri, “Machine Learning Based Phishing Detection from URLs”, Elsevier, vol. 117, pp. 345-357, 2019.

[2]. Routhu Srinivasa Rao, Tatti Vaishnavi, Alwyn Roshan Pais, “CatchPhish: detection of phishing websites by inspecting URLs”, SpringerLink, vol. 11, pp. 813-825, 2019.

[3]. Routhu Srinivasa Rao, Roshan Alwyn Pais, “Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach”, SpringerLink, vol. 11, pp. 3853-3872, 2020.

[4]. Amey Umarekar, Routhu Srinivas Rao, Alwyn Roshan Pais, “Application of word embedding and machine learning in detecting phishing sites”, Telecommunication Systems (Publisher: Springer), vol. 79, 2021.

[5]. Vaibhav Patil, Pritesh Thakkar, Chirag Shah, Tushar Bhat, S P Godse, "Detection and prevention of phishing websites using Machine learning approach" 2018 4th ICCUBEA (Publisher: IEEE), pp. 518-522, 2018.

[6]. T. Natheztha , D Sangeetha, V Vaidehi, "WC-PAD: Web Crawling based Phishing Attack Detection" ,2019 ICCOST(Publisher: IEEE),pp. 1-3, 2019.

[7]. Faan Zheng Qiao Yan, Victor CM Leung, Zhong Ming, “HDP-CNN: Highway deep pyramid convolution neural network combining word-level and character-level representations for phishing website detection”, Computers and Security volume 114, Elsevier, 2022.

[8]. Edwin Raja S, R. Ravi, “A performance analysis of Software Defined Network based prevention on phishing attack in cyberspace using a deep machine learning with CANTINA approach(DMLCA)”, Computer Communications vol 153, Elsevier, 2020.

[9]. Amani Alswailem, Bashayr Alabdullah, Norah Alrumayh, Dr. Aram Alsedrani, " Detecting Phishing Websites using Machine Learning " 2019 2nd ICCAIS (Publisher: IEEE), pp. 821-825, 2019.

[10]. Mohammed Hazim Alkawaz, Stephanie Joanne Steven, Asif Iqbal Hajamydeen, “Detecting Phishing Websites using Machine Learning”, 2020 16th CSPA, IEEE Access, 2020.