IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

PhisherDock

[1]Dr. Dhananjaya V,[2]Anirudh Rai,

[1]Professor and Head, Computer Science and Engineering
[1]Impact College of Engineering and Applied Sciences, Bengaluru, India.

Abstract: Phishing is a fraudulent technique used to extract sensitive data and user credentials by impersonating legitimate websites. Cybercriminals often create duplicate websites with malicious code to steal personal information from unsuspecting users. Such attacks can cause significant financial damage to individuals and businesses using banking and financial services. Traditionally, blacklists of known phishing links or heuristic analysis of suspicious web pages have been used to detect phishing attacks. However, heuristic functions rely on trial and error, resulting in poor accuracy and low adaptability to new phishing links. The primary objective of the project is to develop a machine learning-based solution to identify and block phishing and malicious web links. The aim is to build an advanced software product that employs machine learning algorithms to recognize and flag potentially harmful URLs. This approach will involve leveraging machine learning to overcome these limitations by implementing various classification algorithms and evaluating their performance on our dataset.

IndexTerms—Phishing Detection, Chrome Extension, Machine Learning

I.INTRODUCTION

The rise of internet services has changed the entire way of life. As most services require registration and access, personal information has become necessary to access these personal services. Although cybersecurity technology prevents information from being transmitted and stored in a secure place online, cybercriminals continue to manage and steal personal information in various ways. One of these methods is phishing, which simulates a normal website and tricks users into providing their personal information. Cybersecurity experts have been using technology to block phishing attacks for over a decade, but unfortunately, there are no tools that can completely protect against phishing attacks.

The APWG's Phishing Annual Report shows that phishing attacks have increased over the past year, with more than 100,000 phishing links reported each month. According to the Internet Crime Complaints Center's 2020 annual report, the financial loss from phishing attacks is huge, with losses exceeding $54 million. It is now important to ensure the security of financial services that are easily accessible over the internet. Phishing can cause huge financial losses and is one of the biggest threats to cybersecurity. Many solutions have been proposed to combat phishing, but traditional phishing detection methods have not been successful due to the diversity and evolution of phishing attacks.

As new attack patterns are discovered to stay one step ahead of cybercriminals, it's important to find ways to change phishing detection strategies.

Problem Definition

In the digital age, a significant number of internet users, particularly the elderly, struggle to differentiate between safe and unsafe websites, leading to an increased susceptibility to phishing scams and other malicious attacks. These incidents can result in significant financial losses, often involving the entirety of an individual's life savings. Therefore, it is imperative to develop and implement effective methods to accurately identify safe and unsafe websites. The primary objective of this project is to leverage cutting-edge Machine Learning techniques to detect unsafe websites and provide clear warnings to web surfers to prevent them from falling victim to phishing attacks.

Motivation

In contemporary times, the creation of unmoderated websites is not constrained to any group or entity, thus posing a serious threat to online visitors, leading to substantial financial losses if they fail to distinguish between trustworthy and deceptive websites. To address this issue, PhisherDock has been developed as a tool to intercept and neutralize such fraudulent websites and assist the public in recognizing websites that pose a potential threat to their online safety. The ultimate objective of PhisherDock is to protect individuals and foster a safer online environment.

Goals and Objective

The aim is to address the problem of internet users, especially the elderly, who have difficulty distinguishing between safe and unsafe websites, making them more vulnerable to phishing and other malicious attacks. The objective is to use advanced Machine Learning techniques to identify safe and unsafe websites and prevent web surfers from falling victim to phishing scams. Ultimately, the goal is to reduce the significant financial losses resulting from these attacks.

II. NEED OF THE STUDY

The rapid growth of internet services has brought about significant changes in people's lives. However, it has also led to an increase in cybercrime, including phishing scams and other malicious attacks. The elderly and other vulnerable groups are particularly susceptible to these attacks due to difficulties in identifying safe and unsafe websites. The financial losses resulting from such attacks can be devastating, often involving life savings. Therefore, there is an urgent need to develop and implement effective methods to accurately identify safe and unsafe websites. PhisherDock provides a solution to this problem by leveraging advanced Machine Learning techniques to detect unsafe websites and provide clear warnings to web surfers. The goal of PhisherDock is to prevent web surfers from falling victim to phishing attacks and reduce the significant financial losses that result from such attacks.

III. RESEARCH METHODOLOGY

The methodology section provides an outline of the plan and method used to conduct a study. This includes the universe of the study, sample of the study, data sources, variables, and analytical framework. The details are presented as follows:

3.1 Population and Sample

When developing a Phishing Website Detector, it is important to consider the population, which refers to the total number of identified phishing URLs present on the internet, as well as a representative sample set created for training purposes. Due to the infeasibility of analyzing the entire population, a sample is selected for analysis. This sample must be sufficiently large and representative to yield accurate results. To this end, we collected a sample of publicly available phishing websites from various repositories, including Kaggle, ISCX 2016, Phish Storm, Phish Tank and the 'Phishing Websites Dataset' obtained from the UCI Machine Learning Repository. The size of the sample depends on various factors, such as the availability of data and computational resources. Subsequently, we employed this sample to train and test our machine learning model for detecting phishing websites.

3.2 Data and Sources of Data

The study has utilized a variety of publicly available datasets to conduct comprehensive research on the detection of phishing and legitimate URLs. The datasets used in the analysis include the Phish Storm dataset which consists of 96,018 URLs equally split between legitimate and phishing URLs, the ISCX-URL2016 dataset which includes 35,378 legitimate URLs and 9,965 phishing URLs, and a publicly available Kaggle repository with 35,000 benign URLs. Additionally, the Phish Tank dataset which comprises 400,000 data samples and is regularly updated has been incorporated. Furthermore, the 'Phishing Websites Dataset' from the UCI Machine Learning repository has been incorporated, which consists of 11,055 URLs including 6,157 instances of phishing and 4,898 instances of legitimate websites. These datasets have enabled us to conduct a rigorous and thorough analysis of the effectiveness of various techniques in detecting phishing URLs.

3.3 Theoretical framework

A. Machine Learning

Machine learning is a field of study within artificial intelligence that employs statistical methods to enable computer systems to improve their performance on a given task without being explicitly programmed. Machine learning algorithms have found numerous applications, including but not limited to image and speech recognition, natural language processing, and most importantly, the detection of phishing websites. These algorithms are capable of learning from large datasets, and their performance can improve with more data and training.

The machine learning module assumes a critical role in training and evaluating models within the framework. Specifically, it is responsible for the continual update and refinement of training data sets, and for the automatic initiation of training and testing processes for all models on a regular basis.

This research developed two machine learning models, namely Support vector machines (SVM) and Random Forest.

B. Dataset

The field of machine learning relies heavily on data as its core component. The quality and quantity of data are pivotal factors that dictate the performance of machine learning-based modules. The data collection module serves as the foundation of the system. A data collection task is typically divided into two primary parts: first, the acquisition of data from diverse sources, and second, the analysis and storage of the collected data.

The analysis utilizes several datasets: Phish Storm dataset with 96,018 URLs (50% phishing and 50% legitimate), ISCX-URL2016 dataset with 35,378 legitimate URLs and 9,965 phishing URLs, a Kaggle repository with 35,000 benign URLs, Phish Tank dataset with 400,000 data samples, and UCI Machine Learning repository's 'Phishing Websites Dataset' with 11,055 URLs (6,157 phishing and 4,898 legitimate websites).
The dataset under consideration comprises 30 distinct features, each of which is associated with a specific rule. In the context of this study, a URL is categorized as "phishing" if the associated rule is deemed to be satisfied. Conversely, if the rule is not satisfied, the URL is classified as "legitimate." Each feature in the dataset takes on one of three possible discrete values: a value of "1" indicates that the rule is fully satisfied, a value of "0" suggests partial satisfaction of the rule, while a value of "-1" indicates that the rule is not satisfied at all.

The features contained within the dataset can be classified into three overarching categories:

- **Address Bar -** Employment of IP addresses, extended URLs, abbreviated URLs, utilization of distinctive characters such as "@" or "//", as well as hyphens ("-").

- **Domain** - URL requested, URL embedded within anchor tags, URLs within <meta>, <script>, and <link> tags, Incorporation of "mailto:" within script., The server form handler generates a response of "about:blank".
- **Script** - Employment of unauthorized JavaScript functionalities, pop-ups, and iFrame redirection.

C. Browser Extension

The website provides "alerts" to inform users of the legitimacy of the URLs accessed on each page load. The proposed solution combines a Python-based training stage with a JavaScript-based testing module. The training component has been developed in Python to leverage sophisticated numerical computation libraries. Additionally, the testing stage, which primarily focuses on web-content and feature extraction, involves minimal heavy computation and is thus unlikely to encounter any client-end computation performance issues.

The Chrome extension is developed to comply with Google's standards and comprises three main components: manifest.json, content.js, and background.js. The manifest file includes all the metadata related to the extension and specifies the files and resources associated with it. The content.js file loads on every page in the Chrome browser after the extension deployment. However, it is an unprivileged module that has limited access to DOM elements and requires additional files to interact with external APIs and manipulate the browser user interface. The supporting file, background.js, facilitates message passing to aid the content script with these interactions.

3.4 Algorithms

A. Random Forest

Random forests are a type of classification algorithm that leverages an ensemble of tree predictors. Each tree predictor relies on the values of a randomly sampled vector, and all trees in the forest share the same distribution. The process of constructing a tree involves selecting k « p (the number of variables or features in the training set) as the number of variables to be chosen to determine the decision node. To build the tree, a bootstrap sample is chosen from the n observations in the training set, and the remaining observations are used to estimate the tree's testing error. At a certain node in the tree, k variables are randomly selected as the decision point, and the best split is calculated based on these variables. Unlike other tree algorithms, random forests are always grown and never pruned. They can effectively handle large numbers of variables in a dataset and generate an unbiased estimate of the generalization error during the forest-building process. Additionally, they can estimate missing data well.

However, the random nature of the forest building process poses a challenge for reproducibility, and interpreting the final model and results can be difficult due to the presence of multiple independent decision trees.

B. Support Vector Machines

The Support Vector Machine (SVM) is a type of supervised machine learning discriminative model that adheres to the principle of constructing a separating hyperplane with maximal margin, to minimize the risk of inaccurate predictions.

The Hard-Margin SVM model is formulated for linearly separable data points using the following primal optimization:

$$\begin{aligned} &Objective(min) \quad \frac{1}{2}W^TW \\ &Constraints \quad y_i(W^TX_i + b) >= 1, \quad i = 1, ...n \end{aligned}$$

In this context, the focus is primarily on identifying hyperplanes that satisfy the constraints, and subsequently maximizing the margin only for the hyperplanes that are deemed valid. However, the formulation lacks the capacity to integrate outliers, thereby precluding its applicability to generalization.

In contrast to the Hard-Margin SVM, the Soft-Margin SVM provides the ability to choose the optimal support vectors using the parameter C. The primal objective of the Soft-Margin SVM is outlined below:

$$\begin{aligned} &Objective(min) \quad \frac{1}{2}W^TW + C\sum_{i=1}^{n}\xi_i \\ &Constraints \quad to \quad y_i(W^TX_i + b) >= 1 - \xi_i, \quad i = 1, ...n \\ &\xi_i >= 0, \quad i = 1, ...n \end{aligned}$$

Additionally, data instances that demonstrate non-linear separability are projected onto a higher-dimensional space and subsequently classified utilizing hyperplanes within this augmented space. In this context, the kernel trick assumes a crucial role in reducing computational costs by introducing a kernel function as a substitute for computing higher-dimensional vectors. The kernel functions are:

$$\begin{aligned} &Gaussian \quad K_g(X_i, X_j) = e^{-\frac{||X_i - X_j||^2}{\sigma^2}} \\ &Sigmoid \quad K_s(X_i, X_j) = tanh(\alpha X_i^T X_j + \theta) \\ &Polynomial \quad K_p(X_i, X_j) = (1 + X_i^T X_j)^p \end{aligned}$$

3.5. System Implementation

The proposed implementation attempts to develop a browser extension, utilizing advanced machine learning techniques for the purpose of detecting phishing attacks. Given the benefits of margin flexibility and decreased computational complexity offered by Support Vector Machines (SVM) for classification problems, the implementation employs a persistent model trained by SVM to identify phishing sites. The extension has been designed to specifically support the Chrome browser, owing to its widespread popularity.

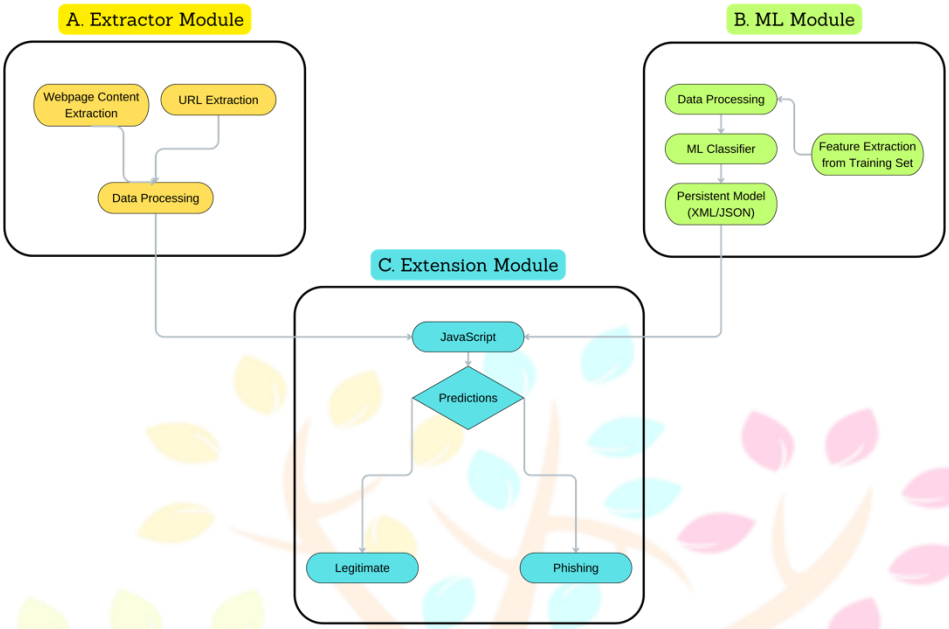

Figure 1. System Implementation

The system implementation of the entire framework is divided into three independent modules: Extractor, ML and Extension.

A. Extractor Module

This module has been designed with multiple functions for web-content and URL feature extraction. The following details explain how phishing portals are identified:

- isIPInURL(): Detects the presence of an IP address in the URL.
- isLongURL(): Validates whether the length of the URL exceeds 75 characters.
- isTinyURL(): Identifies URLs that are shorter than 20 characters.
- isAlphaNumericURL(): Checks for the presence of an alphanumeric '@' in the URL.
- isRedirectingURL(): Verifies if '//' appears more than once within the URL.
- isHyphenURL(): Checks for the presence of '-' adjacent to the domain name in the URL.
- isMultiDomainURL(): Ensures that the domain name is confined to top-level domain, country-code, and second-level domain.
- isFaviconDomainUnidentical(): Verifies if links on the web page are loaded from other domains.
- isIllegalHttpsURL(): Identifies the presence of multiple 'https' in the URL string.
- isImgFromDifferentDomain(): Validates if images on the web page are loaded from other domains.
- isAnchorFromDifferentDomain(): Detects if links on the web page are loaded from other domains.
- isScLnkFromDifferentDomain(): Identifies if scripts on the web page are loaded from other domains.
- isFormActionInvalid(): Detects invalid or blank form submissions.
- isMailToAvailable(): Checks for anchor tags incorporating 'mailto'.
- isStatusBarTampered(): Validates if the onmouseover event manipulates the status bar display.
- isIframePresent(): Identifies sites that exhibit iframes in the DOM.

The extracted features are then passed through an SVM model to identify hostile web-URLs.

B. ML Module

- SCIKIT-LEARN – It is a widely utilized open-source library for predictive data analysis within the field of machine learning. Within the scope of this project, we employed this library to train two traditional machine learning models, specifically the Random Forest and Support Vector Machine (SVM) models.).

- PYTORCH - This is an open-source deep learning framework and development platform. The dataset module was employed to construct a customized dataset as input for the training model. Furthermore, the Torch.cuda package was utilized, which leverages GPUs for parallel computation.

The proposed methodology involves the use of a Support Vector Machine (SVM) discriminative classifier to identify and classify potentially malicious websites. The persistent model generated by the SVM classifier is then passed to a browser

extension, which analyzes the authenticity of the URLs accessed by the user and provides alerts to notify them of any potential threats. The Machine Learning (ML) module is designed as a scalable Web API, which can be consumed by the testing module.

C. Extension Module

This module is designed to integrate Python-based training stage implementation with JavaScript-based testing module. The Python component has been optimized for complex numeric computation libraries. The Chrome extension is compliant with Google standards and consists of three main files: manifest.json, content.js, and background.js. The manifest file contains metadata information about the extension and specifies all associated files and resources. The content.js file loads on every page in the Chrome browser but is an unprivileged module with direct access only to DOM elements, requiring additional files to interact with external APIs and manipulate the browser user interface. The background.js file supports content script interactions through message passing. Development of the Chrome browser extension must strictly follow development guidelines, including version number control, built-in APIs introduction, and permission control. Data interactions between modules are messaged and temporarily stored in Chrome storage. A popup HTML interface displays the warning.

3.6 Data Collection and Cleaning

The datasets used in the analysis include the Phish Storm dataset which consists of 96,018 URLs equally split between legitimate and phishing URLs, the ISCX-URL2016 dataset which includes 35,378 legitimate URLs and 9,965 phishing URLs, and a publicly available Kaggle repository with 35,000 benign URLs. Additionally, the Phish Tank dataset which comprises 400,000 data samples and is regularly updated has been incorporated. Furthermore, the 'Phishing Websites Dataset' from the UCI Machine Learning repository has been incorporated, which consists of 11,055 URLs including 6,157 instances of phishing and 4,898 instances of legitimate websites. Given that the data has already undergone a thorough cleaning process, there was no necessity for any additional cleansing procedures.

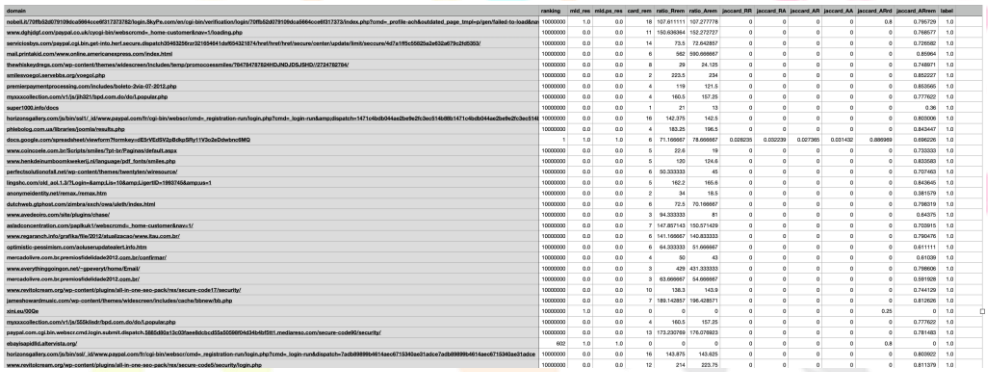

Figure 2. Sample Of Phishing Data (Phish Storm)

3.7 Data Pre-Processing

Data pre-processing is a crucial initial phase in any machine learning endeavor, such as the detection of phishing websites. This step involves transforming and cleaning raw data to prepare it for analysis. In the case of detection of phishing websites, data preprocessing entails a range of tasks, such as eliminating extraneous data, handling missing values, and formatting data to a machine-readable form.

Overall, data pre-processing is a vital step that ensures the data is prepared for analysis, and machine learning algorithms can accurately identify phishing websites.

3.8 Model Training

- SCIKIT-LEARN – It is a widely utilized open-source library for predictive data analysis within the field of machine learning. Within the scope of this project, we employed this library to train two traditional machine learning models, specifically the Random Forest and Support Vector Machine (SVM) models.).

- PYTORCH - This is an open-source deep learning framework and development platform. The dataset module was employed to construct a customized dataset as input for the training model. Furthermore, the Torch.cuda package was utilized, which leverages GPUs for parallel computation.

IV. RESULTS AND DISCUSSION

4.1 Confusion Matrix

	Predicted Phishing URLs	Predicted Legitimate URLs
Ground Truth Phishing URLs	1249	162
Ground Truth Legitimate URLs	182	1680

Figure 3. Random Forest Confusion Matrix

	Predicted Phishing URLs	Predicted Legitimate URLs
Ground Truth Phishing URLs	1293	206
Ground Truth Legitimate URLs	131	1731

Figure 4. SVM Confusion Matrix

In the field of detecting phishing websites, the evaluation of a model's classification performance is often conducted using a

confusion matrix, which allows for a comprehensive analysis of the model's true positive, true negative, false positive, and false negative rates. In the present study, we have conducted an analysis of the performance of two different models, Random Forest and Support Vector Machines (SVM), using their respective confusion matrices.

The results indicate that the SVM model performed with high accuracy in accurately identifying legitimate and phishing websites, as evidenced by its higher number of predicted phishing URLs (1293) compared to the Random Forest model (1249). This indicates that the SVM model was more effective in minimizing false positives and maximizing true positives, resulting in more accurate detection of phishing URLs. Overall, these findings suggest that SVM may be a more suitable model for detecting phishing websites and could be further developed for use in real-world applications.

	Accuracy(%)	Specificity(%)	Sensitivity(%)
SVM	89.84	93	89
Random Forest	89.63	90	86

Figure 5. Performance Matrix of Classifiers

4.2 Result

The implementation of a phishing website detection model as a Chrome Extension Plug-in is shown in Figure 6.

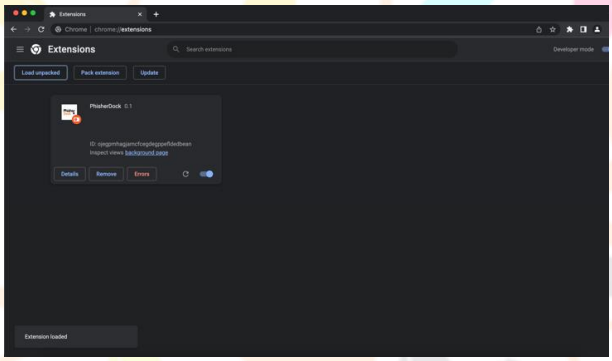

Figure 6. Loading PhisherDock as Chrome Extension

Figure 7 shows a legitimate website (www.google.com) being detected and a pop-up showing “No Phishing Detected” being displayed.

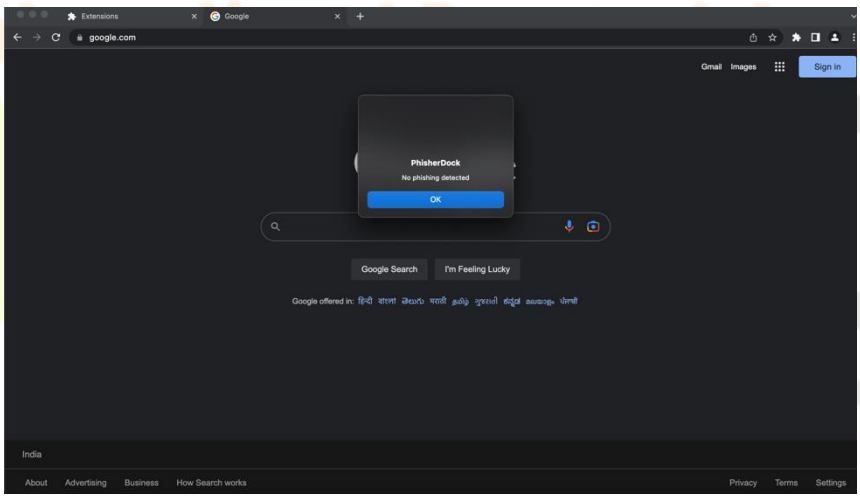

Figure 7. Legitimate Website Detected

Figure 8 shows a phishing website (http://pestanaleilaoficial.com/) being detected and a pop-up showing “Warning: Phishing Detected!!” being displayed.

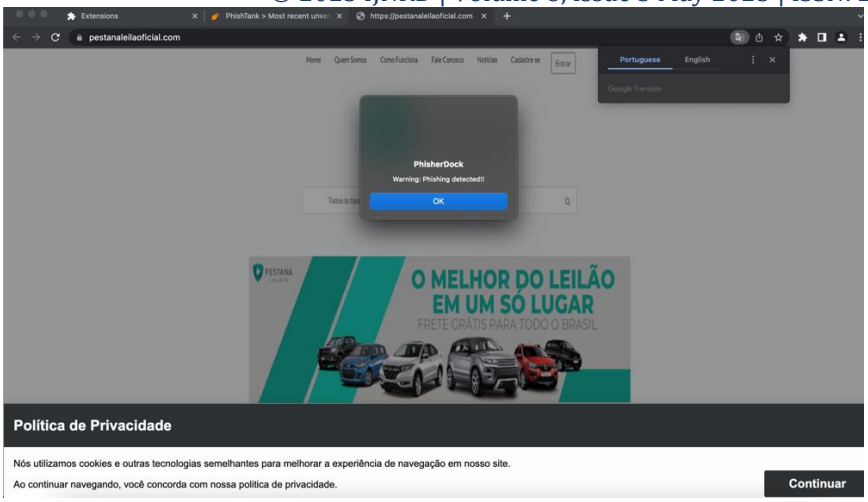

Figure 8. Phishing Website Detected

Figure 9 shows PhisherDock being easily accessible in the toolbar.

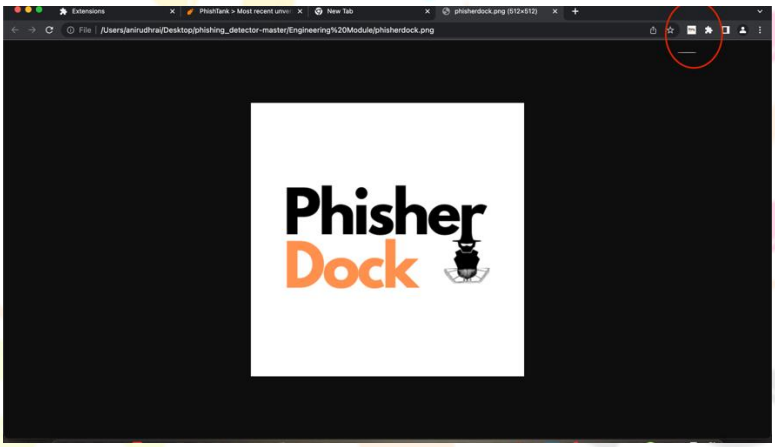

Figure 9. PhisherDock Accessible in the Tool Bar

V. ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the Project would be incomplete without the mention of the people who made it possible, without whose constant guide and encouragement would have made our efforts go in vain. We consider ourselves privileged to express our gratitude and respect towards all those who guided us through the completion of this Project on, "PhisherDock".

We consider ourselves proud to be part of Impact College of Engineering and Applied Sciences, Bangalore, the institute which stood by us through our endeavours. We would like to thank our guide Dr. Dhananjaya V, Professor and Head, Department of Computer Science and Engineering, for the support and encouragement provided throughout the process.

REFERENCES

[1] S. Maurya, H. Singh, and A. Jain, ''Browser extension based hybrid anti- phishing framework using feature selection,'' Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 11, pp. 1–10, 2019, doi: 10.14569/ijacsa.2019.0101178.

[2] B. Shah, K. Dharamshi, M. B. Patel, and V. Gaikwad. (2020). Chrome Extension for Detecting Phishing Websites. Semantic Scholar. [Online]. Available: https://www.semanticscholar.org/paper/Chrome- Extension-for-Detecting-Phishing-Websites-Shah- Dharamshi/fa99621bdc27cbd32ed799d7a6c1848ac644e8a8

[3] O. Abiodun, A. S. Sodiya, and S. O. Kareem, ''Linkcalculator—An efficient link-based phishing detection tool,'' Acta Inf. Malaysia, vol. 4, no. 2, pp. 37–44, Oct. 2020, doi: 10.26480/aim.02.2020.37.44.

[4] M. G. Hr, M. V. Adithya, and S. Vinay, ''Development of anti-phishing browser based on random forest and rule of extraction framework,'' Cyber- security, vol. 3, no. 1, pp. 1–14, Oct. 2020, doi: 10.1186/s42400-020- 00059-1.

[5] D. N. Atimorathanna, T. S. Ranaweera, R. A. H. Devdunie Pabasara, J. R. Perera, and K. Y. Abeywardena, ''NoFish; total anti-phishing pro- tection system,'' in Proc. 2nd Int. Conf. Advancements Comput. (ICAC), Dec. 2020, pp. 470–475, doi: 10.1109/ICAC51239.2020.9357145.

[6] K. M. Sundaram, R. Sasikumar, A. S. Meghana, A. Anuja, and C. Praneetha, ''Detecting phishing websites using an efficient featur e- based machine learning framework,'' Revista Gestão Inovação e Tecnologias, vol. 11, no. 2, pp. 2106–2112, Jun. 2021, doi: 10.47059/revistagein- tec.v11i2.1832.

[7] Lizhen Tang, Qusay H. Mahmoud, "A Deep Learning-Based Framework for Phishing Website Detection," in Proc. IEEE, 2022