

PhishFinder: A Real-Time Phishing Website Detection System

*Phase II project report submitted in partial fulfilment of the requirements for
the award of the degree of*

Bachelor of Technology

in

Computer Science & Engineering

Submitted by

Joyal Devassy

Nikhil Ravi

Vijay Bhaskar

Vishal C Varghese



Focus on Excellence

Federal Institute of Science And Technology (FISAT)®
Angamaly, Ernakulam

Affiliated to

APJ Abdul Kalam Technological University
CET Campus, Thiruvananthapuram
May 2024

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY
(FISAT)

Mookkannoor(P.O), Angamaly-683577



CERTIFICATE

This is to certify that the project phase II report for the project “**PhishFinder: A Real-Time Phishing Website Detection System**” is a bonafide report of the project presented during VIIIth semester (CSD416 - Project Phase II) by **Nikhil Ravi (FIT20CS087)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B.Tech) in Computer Science & Engineering during the academic year 2023-24.

Ms. Hansa J Thattil	Ms. Sruthy Suresh	Dr. Jyothish K John
Project Coordinator	Project Guide	Head of the Department

ABSTRACT

The proliferation of phishing websites poses a significant threat to users' online security. In response, we propose a browser extension designed to detect 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 intervention 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 contributing to its classification. For safe websites, the extension displays the safety level, offering users confidence in their browsing experience. This browser extension empowers users to navigate the web securely, mitigating the risks associated with phishing attacks through proactive detection and informative alerts.

Contribution by Author

Design and implement the user interface for the browser extension, including the main dashboard, real-time alerts, and interactive features. Develop the browser extension's front end using web technologies like HTML, CSS, and JavaScript, ensuring compatibility across different browsers. Implement the functionality for displaying real-time alerts upon detecting potential phishing websites, providing seamless user experience without requiring user intervention. Create interactive components, such as the extension icon, tooltips, and detailed information popup, allowing users to access safety level and feature analysis of the current website. Conduct usability testing and iterate on the design based on user feedback to enhance user satisfaction and adoption.

Nikhil Ravi

ACKNOWLEDGMENT

We would like to express our utmost gratitude to , Principal, Federal Institute of Science and Technology, Angamaly. We are fortunate to be blessed with the guidance of **Dr. Jyothish K John**, Head of Department, Computer Science and Engineering, FISAT, Angamaly. It gives us immense pleasure to express our sincere gratitude to **Ms. Sruthy Suresh(Project Guide)** and **Ms. Hansa J Thattil(Project Coordinator)** for their valuable and untiring guidance and supervision throughout the tenure of our work. To derive benefits from their numerous experience is a matter of great privilege for us. We also take this opportunity to express our sincere thanks to all the staff in the computer science department, who extended their wholehearted cooperation, moral support, and rendered ungrudging assistance whenever and wherever the need arose.

Contents

List of Figures

List of Tables

1	Introduction	1
1.1	Overview	1
1.2	Problem Statement	1
1.3	Objective	2
1.4	Scope of the project	2
1.5	Proposed Work	2
1.6	Organization of the report	3
2	Literature Review	4
2.1	Literature survey	4
2.1.1	PDGAN: Phishing Detection With Generative Adversarial Networks[8]	4
2.1.2	PhiKitA: Phishing Kit Attacks Dataset for Phishing Websites Identification[10]	5
2.1.3	Particle Swarm Optimization-Based Feature Weighting for Improving Intelligent Phishing Website Detection[5]	5
2.1.4	Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism[4]	6
2.1.5	Phishing URL Detection: A Real-Case Scenario Through Login URLs[7]	6
2.1.6	SPWalk: Similar Property Oriented Feature Learning for Phishing Detection[9]	6
2.1.7	The Answer Is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering[6]	7
2.1.8	A Boosting-Based Hybrid Feature Selection and Multi-Layer Stacked Ensemble Learning Model to Detect Phishing Websites[3]	7
2.1.9	Web2Vec: Phishing Webpage Detection Method Based on Multidimensional Features Driven by Deep Learning[2]	8
2.1.10	Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML[1]	8
2.3	Proposed System	10
2.2	Comparison Table	11
3	Design	13
3.1	Functional Requirements	13
3.2	Non-Functional Requirements	13
3.3	Design Methodologies	14

3.4	Data Flow Diagram	16
4	Implementation	17
4.1	Implementation Details	17
4.1.1	Dataset Preprocessing	17
4.1.2	Training	18
4.1.3	Exporting Model	18
4.1.4	Plugin Feature Extraction	18
4.1.5	Exporting_Algorithm	20
5	Results	21
5.1	Training	21
5.2	Plugin Feature Extraction	21
5.3	Classification	22
5.4	Social Relevance	22
5.5	Future Scope	23
6	Conclusion	25
References		
Appendices		i
A	CODE	ii
A.0.1	Machine Learning Model	ii
B	Screenshots	iv

List of Figures

3.1	System Architecture	15
3.2	Data Flow Diagram	16
5.1	Phishing Website	22
B.1	Legimate Website	iv
B.2	Phishing Website	v
B.3	Parameters	vi
B.4	Warning	vii

List of Tables

2.1	Comparison of Phishing Detection Models	11
2.2	Comparison of Literature Papers	12

Chapter 1

Introduction

1.1 Overview

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.

1.2 Problem Statement

Phishing, a prevalent cyber threat, targets sensitive user information through fraudulent means, often via spoofed emails or websites. Traditional methods like directory lookups struggle with the transient nature of phishing sites. Machine learning, particularly the random forest classifier, shows promise in detection. Existing browser plugins rely on external servers for classification, raising privacy concerns and latency issues. Our solution is a Chrome plugin that performs real-time classification locally, preserving user privacy and ensuring instant warnings upon encountering phishing sites. This project aims to empower users with timely alerts without compromising their browsing data, enhancing overall cybersecurity.

1.3 Objective

The primary objectives of Phishfinder can be summarized as follows:

- Develop a Chrome browser plugin for real-time phishing detection
- Ensure the plugin functions independently without relying on external servers.
- Provide instant warnings to users upon encountering phishing websites.
- Preserve user privacy by avoiding transmission of browsing data to external sources.
- Enhance overall cybersecurity by empowering users with timely alerts during web browsing.

1.4 Scope of the project

The scope of the project involves developing a browser extension that utilizes machine learning, specifically a Random Forest model, to analyze website URLs in real-time for potential phishing threats. The extension automatically alerts users when it detects a suspicious website, providing a seamless browsing experience. Additionally, it offers users the ability to interact with the extension for detailed information about website safety levels and the features contributing to their classification. Overall, the project aims to empower users to navigate the web securely by proactively detecting and informing about phishing threats.

1.5 Proposed Work

The project aims to develop a browser extension to combat the rising threat of phishing websites. Leveraging machine learning, specifically a Random Forest model trained on diverse URL features, the extension will analyze website URLs in real-time as users browse the web. Upon detecting potential phishing sites, it will automatically display warning messages, ensuring enhanced ease of use without requiring user intervention. Users can also interact with the extension by clicking its icon, providing detailed safety information and classification features of the current website. For safe websites, the extension instills confidence by displaying safety levels.

This proactive approach empowers users to navigate securely, mitigating the risks associated with phishing attacks through timely alerts and informative insights. The proposed work involves comprehensive planning, including research on existing methods, data collection, and preprocessing. Development encompasses machine learning model creation, frontend and backend implementation, and thorough testing. Documentation and deployment ensure usability and effectiveness, with ongoing monitoring for continuous improvement. Through this project, users will navigate the web with increased security and peace of mind.

1.6 Organization of the report

- Chapter 2 consists of a literature review of 10 research papers taken for the project. Each paper is explained based on its methodology and results. A comparison of each paper is done in this section.
- Chapter 3 outlines the design of the system. It explains how the functional and non-functional requirements, methodologies used, and various diagrams representing the implementation. It's like the blueprint for our project.
- Chapter 4, is a detailed implementation of the entire project. Dataset, Libraries, and applications used are specified.
- Chapter 5, wrapped everything up. It looks back at what we've achieved and discusses why our project is important. It's like the conclusion of a story, summarizing our journey and highlighting the impact of our work in the early detection of autism spectrum disorders.

Chapter 2

Literature Review

2.1 Literature survey

Phishing attacks have become increasingly prevalent, posing a significant threat to users' online security. In response to this growing problem, researchers and developers have been exploring various methods to detect and prevent phishing attempts. One promising approach involves the use of machine learning techniques to analyze website URLs and identify potential phishing websites in real-time. This literature review will examine the existing research and technologies in this field, focusing on the use of machine learning for phishing website detection in browser extensions.

2.1.1 PDGAN: Phishing Detection With Generative Adversarial Networks[8]

The paper addresses the critical challenge of phishing attacks, emphasizing the urgent need for highly accurate detection tools amidst the proliferation of online services. Conventional methods often suffer from high false detection rates, prompting exploration into deep learning techniques, particularly generative adversarial networks (GANs). The proposed model, PDGAN, introduces a unique approach by focusing exclusively on a website's URL for detection. It employs a dual-network architecture consisting of LSTM and CNN, trained on a dataset of approximately two million URLs sourced from PhishTank and DomCop. Impressively, PDGAN achieves a detection accuracy of 97.58% and precision of 98.02%, outperforming existing models while operating independently of third-party services. Furthermore, the paper's contributions extend to the development of a phishing detection extension, which leverages PDGAN's capabilities. By solely relying on URL-based detection, the extension mitigates privacy concerns and reduces false positives associated with content-based methods. This approach aligns well with the efficiency goals of browser extensions, promising improved security in online environments. Overall, the study presents a significant advancement in phishing detection, offering high accuracy and independence from external sources, thus paving the way for more robust cybersecurity measures.

2.1.2 PhiKitA: Phishing Kit Attacks Dataset for Phishing Websites Identification[10]

PhiKitA, a pioneering dataset, combats the rising threat of phishing attacks facilitated by phishing kits. These kits enable phishers to execute schemes with alarming efficiency and scale. The lack of comprehensive datasets cataloging these tools and infiltrated websites impedes early detection efforts. PhiKitA addresses this gap by encompassing phishing kits and their spawned websites. Employing MD5 hashes, fingerprints, and innovative graph representation DOM algorithms, our methodology conducts three pivotal experiments. Firstly, our familiarity analysis delves into phishing kit characteristics, shedding light on diversity and emerging campaigns. Secondly, we tackle phishing website detection, achieving a 92.50% accuracy rate with our graph representation algorithm, highlighting its efficacy. Lastly, we identify the origins of phishing websites by scrutinizing unique signatures and structures within phishing kits, revealing connections to fraudulent sources. While our graph representation excels in detection, the MD5 hash approach yields a respectable F1 score of 39.54%, showcasing its utility in specific contexts. PhiKitA empowers researchers and cybersecurity professionals with a comprehensive dataset, enabling the development of more effective detection and mitigation strategies. Ultimately, this bolsters online ecosystem resilience against malicious actors, marking PhiKitA as a pivotal resource in the fight against phishing threats.

2.1.3 Particle Swarm Optimization-Based Feature Weighting for Improving Intelligent Phishing Website Detection[5]

The paper introduces a novel approach to combat web phishing attacks using particle swarm optimization (PSO) for feature weighting in a phishing website detection system. Traditional methods struggle to keep up with evolving phishing tactics, particularly zero-day threats. PSO dynamically adjusts feature weights based on their effectiveness in distinguishing phishing from legitimate websites, improving accuracy. The methodology involves collecting a dataset with various website features, splitting it into training and testing sets. Machine learning models, like decision trees or support vector machines, are then employed to build the detection system using optimized feature weights. The system's performance is evaluated using metrics like classification accuracy and false positive rates. Results indicate significant improvements in detection accuracy and efficiency, even with a reduced set of features. By autonomously prioritizing and adjusting feature weights, the system effectively identifies phishing websites, enhancing cybersecurity measures and bolstering trust in online services.

2.1.4 Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism[4]

The paper introduces THEMIS, a novel phishing email detection model leveraging an advanced recurrent convolutional neural networks (RCNN) architecture with multilevel vectors and an attention mechanism. THEMIS analyzes email headers, bodies, character-level, and word-level features simultaneously to enhance phishing email detection accuracy. Evaluation on a realistic dataset demonstrates an impressive 99.848% overall accuracy with a low 0.043% false positive rate (FPR). Integrating THEMIS into phishing website detection systems enhances cybersecurity by indirectly aiding in identifying and mitigating associated malicious websites. Its nuanced understanding of phishing email patterns improves accuracy in recognizing sophisticated phishing techniques. The paper's emphasis on using a realistic dataset ensures THEMIS' practicality and robustness in real-world cybersecurity applications. With exceptional performance and adaptability, THEMIS emerges as a promising solution for addressing the escalating threat of phishing attacks and enhancing overall cybersecurity measures.

2.1.5 Phishing URL Detection: A Real-Case Scenario Through Login URLs[7]

In this paper, the authors address the challenge of phishing detection by comparing machine learning and deep learning techniques, specifically focusing on URL analysis. Unlike prevailing methods that define the legitimate class solely based on homepages and exclude login forms, the proposed approach considers URLs from both homepages and login pages for both classes. This adjustment aims to better reflect real-world scenarios and exposes the high false-positive rates associated with existing techniques when confronted with URLs from legitimate login pages. The study also investigates the temporal aspect of model accuracy by training on old datasets and testing on recent URLs, highlighting a decline in performance over time. The authors introduce a new dataset, Phishing Index Login URL (PILU-90K), consisting of 60K legitimate URLs and 30K phishing URLs. A Logistic Regression model, coupled with Term Frequency - Inverse Document Frequency (TF-IDF) feature extraction, achieves an impressive 96.50% accuracy on the presented login URL dataset, underscoring the effectiveness of the proposed method in detecting phishing websites. Additionally, the paper conducts a frequency analysis of current phishing domains, shedding light on various techniques employed by phishers in their campaigns.

2.1.6 SPWalk: Similar Property Oriented Feature Learning for Phishing Detection[9]

SPWalk is a cutting-edge algorithm designed to tackle the growing threat of phishing attacks by analyzing the complex network of weblinks between webpages. By

employing a biased random walk procedure, it maps these webpages into a lower-dimensional vector space, effectively capturing both structural relationships and URL details. This integration allows SPWalk to effectively discern phishing attempts from legitimate webpages. Three key strengths underpin SPWalk's effectiveness: firstly, the inherent difficulty attackers face in fully controlling reference relationships, which makes it harder for them to mimic legitimate webpages. Secondly, SPWalk leverages the structural regularities emerging from diverse reference relationships to differentiate between phishing and legitimate webpages. Lastly, by incorporating node URL information into its analysis, SPWalk enhances the accuracy of its classifications. Experimental evaluations consistently demonstrate SPWalk's superiority over existing techniques, particularly in achieving precision rates exceeding 95%. Its robustness in identifying phishing attempts, even in scenarios where attackers employ sophisticated evasion tactics, positions SPWalk as a highly promising solution for combatting the evolving landscape of web-based security threats.

2.1.7 The Answer Is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering[6]

This project proposes a multi-stage methodology for detecting phishing email attacks, integrating natural language processing (NLP) and machine learning techniques. It begins with feature engineering, lemmatization, and dimensionality reduction using Chi-Square statistics, Mutual Information, PCA, and LSA. By addressing issues like sparsity and the curse of dimensionality, the method achieves an impressive 100% F1-measure in validation tests using reputable datasets. The approach's significance lies in its ability to accurately identify phishing emails while minimizing computational costs. It represents a notable advancement in cybersecurity, particularly in combating email-based phishing threats. By leveraging advanced NLP techniques and machine learning algorithms, it provides a robust framework for distinguishing between phishing and legitimate emails. Moreover, the methodology's thoroughness, incorporating feature selection and fine-tuning of machine learning models, ensures robustness and generalizability. Its success in achieving perfect F1-measure underscores its effectiveness in accurately detecting phishing attempts. Overall, this multi-stage approach offers a promising solution for enhancing email security and mitigating the risks posed by phishing attacks.

2.1.8 A Boosting-Based Hybrid Feature Selection and Multi-Layer Stacked Ensemble Learning Model to Detect Phishing Websites[3]

The paper introduces an innovative approach to tackle phishing attacks using a boosting-based multi-layer stacked ensemble learning model. This model incorporates hybrid feature selection to identify pertinent features for classification. It operates in multiple layers, with each layer employing different classifiers, and utilizes predictions from lower layers to enhance detection accuracy. Experimental results demonstrate consistently high accuracy levels ranging from 96.16% to 98.95%

without feature selection and 96.18% to 98.80% with feature selection across various datasets. Compared to baseline models, the proposed approach significantly outperforms existing methods. Key contributions include the novel model architecture, integration of boosting and ensemble learning, and effective feature selection technique. These contributions enhance the model's ability to discern subtle phishing patterns and improve overall detection performance. In summary, the paper presents a sophisticated strategy for combating phishing attacks, offering a promising solution to the ongoing threat posed by deceptive online practices.

2.1.9 Web2Vec: Phishing Webpage Detection Method Based on Multidimensional Features Driven by Deep Learning[2]

The paper introduces two innovative methodologies to enhance phishing website detection. The first method employs representation learning and a hybrid deep learning network to automatically extract features from diverse aspects of phishing webpages. This approach addresses challenges related to manual feature collection and correlation issues. By incorporating an attention mechanism, the model effectively prioritizes critical features, resulting in impressive accuracy of 99.05% and a remarkably low false positive rate of 0.25%. The second approach introduces a boosting-based multi-layer stacked ensemble learning model, which utilizes a hierarchical architecture to capture intricate phishing patterns. Additionally, a hybrid feature selection technique optimizes the model's focus on discriminative features, reducing overfitting risks. Experimental analysis demonstrates consistent high accuracy, ranging from 96.16% to 98.95%, across various datasets. Moreover, comparative assessments against baseline models underscore the superior performance of the proposed methodologies. These contributions signify significant advancements in phishing detection by leveraging sophisticated techniques to enhance model efficacy and robustness, ultimately improving cybersecurity measures against malicious online activities.

2.1.10 Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML[1]

The paper introduces two innovative approaches to phishing webpage detection: PhishNet and PhishDet. PhishNet leverages representation learning and a hybrid deep learning network to automatically extract features from various aspects of webpages. This approach addresses challenges such as feature complexity and correlation issues between features, achieving notable accuracy and a low false positive rate. By incorporating attention mechanisms and fusing outputs from multiple channels, PhishNet demonstrates effectiveness in phishing detection. On the other hand, PhishDet stands out by integrating Long-term Recurrent Convolutional Network (LRCN) and Graph Convolutional Network (GCN) architectures. This allows for comprehensive analysis of sequential data from URLs and relational patterns within HTML content. The incorporation of GCN

enables a deeper understanding of complex relationships within webpage structures, significantly improving detection accuracy. PhishDet also boasts impressive performance metrics, including high detection accuracy, low false-negative rates, and swift response times, highlighting its practical impact in countering phishing threats. Moreover, PhishDet's utilization of automatic feature selection enhances adaptability to emerging threats, ensuring robustness in real-world scenarios. This adaptive learning capability is crucial for handling the constantly evolving nature of phishing attacks. The paper underscores the importance of periodic retraining to maintain performance over time, yet emphasizes PhishDet's consistency.

2.2 Comparison Table

Paper Name	Model Used	Accuracy	Features & Novelty
Long-Term Recurrent Convolutional and Graph Convolutional Networks[1]	Long-term Recurrent Convolutional Network and Graph Convolutional Network	96.42%	Uses URL and HTML features. Claims to be the first to use Graph Neural Network in anti-phishing with high accuracy. Requires periodic retraining.
Web2Vec: Phishing Webpage Detection Method Based on Multi-Dimensional Features Driven by Deep Learning [2]	Hybrid deep learning network with CNN and bidirectional LSTM	99.05%	Utilizes URL, HTML page content, and DOM structure. Emphasizes representation learning from multi-aspect features.
A Boosting-Based Hybrid Feature Selection and Multi-Layer Stacked Ensemble Learning Model to Detect Phishing Websites[3]	Boosting based multi-layer stacked ensemble learning	96.16%-98.95% without feature selection, and 96.18%-98.80% with feature selection	Hybrid feature selection technique. Uses boosting and ensemble learning for phishing detection.
Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism [4]	Improved recurrent convolutional neural networks (RCNN) with attention mechanism	99.848% overall accuracy with a low false positive rate (FPR) of 0.043%	Considers email header, body, character level, and word level simultaneously. Uses multilevel vectors and attention mechanism in RCNN.
Particle Swarm Optimization-Based Feature Weighting for Improving Intelligent Phishing Website Detection [5]	Particle Swarm Optimization (PSO)-based feature weighting	Outstanding improvements in classification accuracy	Uses PSO to weight various website features effectively. Focuses on intelligent phishing website detection using PSO-based feature weighting.

Table 2.1: Comparison of Phishing Detection Models

Paper Name	Model Used	Accuracy	Features & Novelty
The Answer Is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering[6]	Logistic Regression with TF-IDF feature extraction	96.50% accuracy on the introduced login URL dataset	Uses URLs from login pages in both classes for training. Considers the inclusion of login pages in the legitimate class.
Phishing URL Detection: A Real-Case Scenario Through Login URLs[7]	XGBoost and Random Forest with multi-stage approach	F1-measure of 100% success rate	Involves feature engineering, lemmatization, and feature extraction within natural language processing. Proposes a multi-stage approach with Chi-Square statistics, Mutual Information, PCA, and LSA.
PDGAN: Phishing Detection With Generative Adversarial Networks[8]	PDGAN model using GAN, LSTM, and CNN	97.58% detection accuracy	Depends only on the website's URL. Proposes a phishing detection model based on GAN without relying on webpage content.
SPWalk: Similar Property Oriented Feature Learning for Phishing Detection[9]	SPWalk - an unsupervised feature learning algorithm	Demonstrates superiority over state-of-the-art techniques, especially in terms of precision	Constructs a weblink network with structural information between nodes and URL information. Focuses on a network embedding technique for feature extraction.
PhiKitA: Phishing Kit Attacks Dataset for Phishing Websites Identification[10]	combination of MD5 hashes, fingerprints, and graph representation DOM algorithms	92.50%	The project offers comprehensive phishing kit analysis using MD5 hashes and graph representation for early website detection, providing a novel approach to cybersecurity.

Table 2.2: Comparison of Literature Papers

2.3 Proposed System

- **Browser Extension Installation:** Users install the browser extension from the appropriate extension store (e.g., Chrome Web Store, Mozilla Add-ons).
- **Real-time URL Analysis:** The extension continuously monitors the URLs users visit while browsing the web.
- **Feature Extraction:** When a URL is accessed, the extension extracts various features from it, such as domain name, subdomain, length of URL, presence of special characters, etc.
- **Machine Learning Model:** These features are fed into a pre-trained machine learning model, such as a Random Forest classifier. The model has been trained on a dataset containing both legitimate and phishing URLs, learning to distinguish between the two based on the extracted features.
- **Phishing Detection:** The model predicts the likelihood of the accessed URL being a phishing website. If the predicted likelihood exceeds a certain threshold, the extension considers the website potentially malicious.
- **User Alert:** If a potential phishing website is detected, the extension displays a warning message to the user, alerting them to the potential threat. The warning message might contain information about why the website is flagged as potentially malicious and advise the user to proceed with caution or avoid the website altogether.
- **User Interaction:** The user can choose to ignore the warning and continue to the website or take precautions based on the alert provided. They can also access more detailed safety information by clicking on the extension icon, which might include tips for identifying phishing websites and best practices for online security.

Chapter 3

Design

3.1 Functional Requirements

- **Real-Time Phishing Detection:**
The extension must analyze website URLs in real-time as users navigate the web. It should leverage a Random Forest model to detect potential phishing websites based on various extracted features. Upon detecting a potential phishing website, the extension must display a warning message automatically without user intervention.
- **User Interaction:**
Users should be able to interact with the extension by clicking on its icon. Clicking on the icon should provide detailed information about the safety level of the current website. The extension should display the extracted features contributing to the website's classification as phishing or safe.
- **User Interface:**
The extension must have an intuitive and user-friendly interface. It should display clear and informative warning messages and safety levels to users. Interactive components such as icons, tooltips, and popups should be visually appealing and easy to understand.
- **Customization Options:**
Users should have the option to customize alert thresholds and notification preferences. The extension may provide settings for users to adjust the sensitivity of phishing detection or disable certain features if desired.

3.2 Non-Functional Requirements

- **Performance:**
The extension must have low latency in analyzing website URLs and displaying alerts. It should be able to handle a high volume of website requests without significant performance degradation.
- **Scalability:**
The backend infrastructure should be scalable to accommodate an increasing number of users and website requests. Load balancing and caching mechanisms should be implemented to distribute workload efficiently.

- **Accuracy:**
The Random Forest model used for phishing detection must achieve high accuracy, precision, recall, and F1-score. False positive and false negative rates should be minimized to ensure reliable detection.
- **Security:**
The extension must handle user data securely and protect against potential security threats. Communication between the frontend and backend should be encrypted to prevent data interception. The backend infrastructure should be resilient to attacks such as SQL injection and cross-site scripting (XSS).
- **Compatibility:**
The extension should be compatible with major web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge. It should work seamlessly across different operating systems and browser versions.

3.3 Design Methodologies

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. With the encoded feature vector and model JSON, the script can run the classification. Then a warning is displayed to the user, incase the website is classified as phishing. The entire system is designed lightweight so that the detection will be rapid.

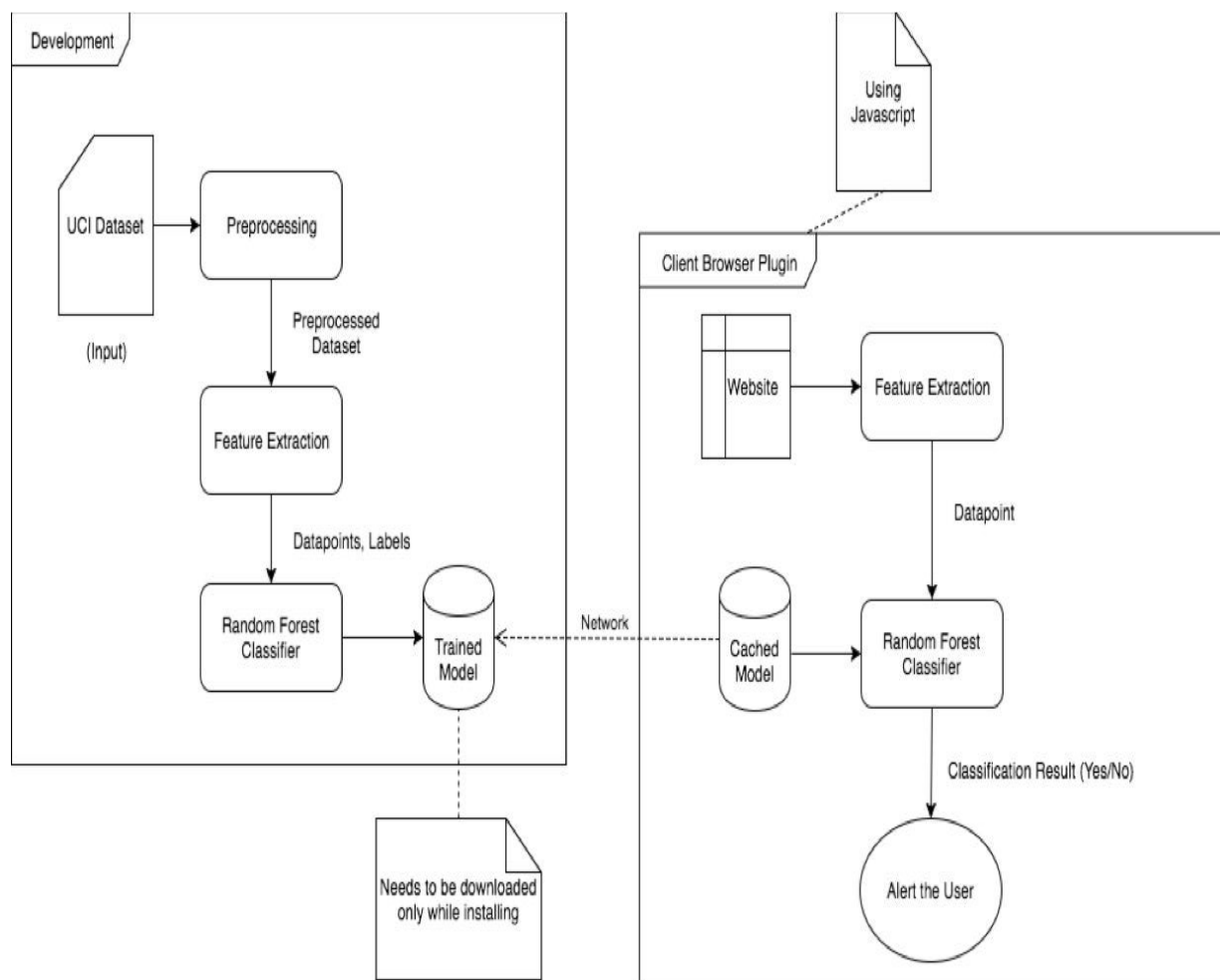


Figure 3.1: System Architecture

3.4 Data Flow Diagram

The project's dataflow facilitates seamless interaction between user browsing, the extension, and the backend model. Users navigate the web, triggering URL extraction and analysis by the extension. Extracted features undergo evaluation by a Random Forest model to detect potential phishing sites. Real-time alerts are automatically displayed, enhancing user security without intervention. Alternatively, users can access detailed website safety information by interacting with the extension icon. Customization options enable users to tailor alert thresholds. Continuous learning mechanisms in the backend ensure model adaptation to evolving threats. This streamlined dataflow empowers users to navigate the web securely, mitigating phishing risks effectively.

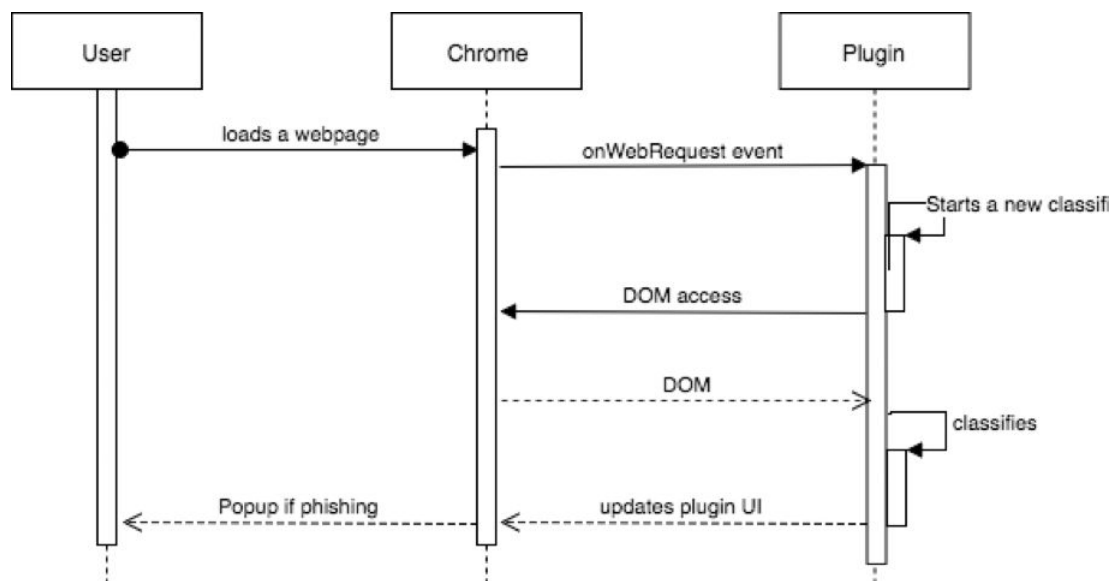


Figure 3.2: Data Flow Diagram

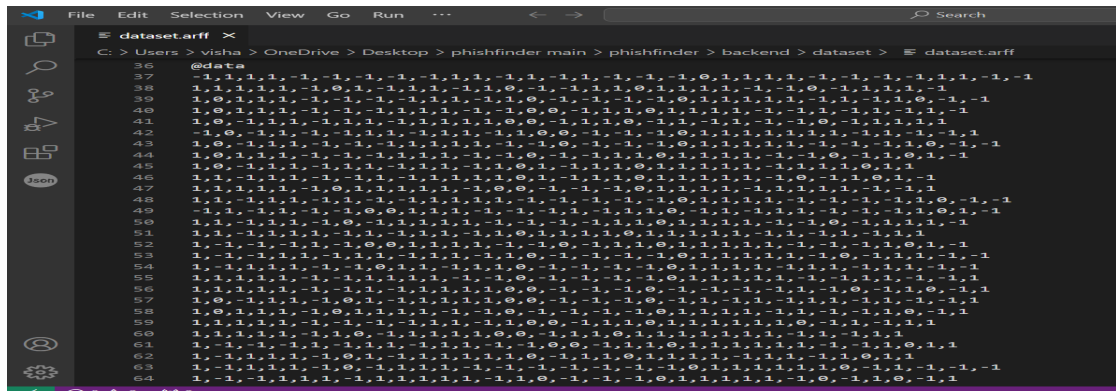
Chapter 4

Implementation

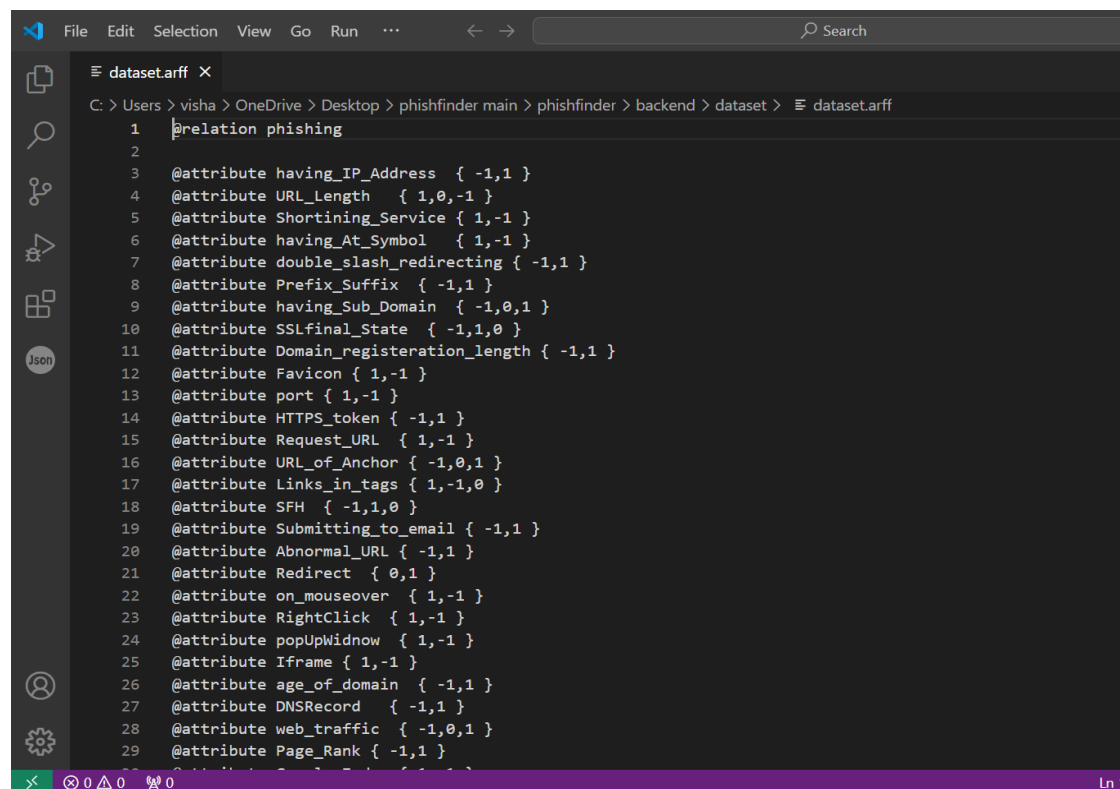
4.1 Implementation Details

4.1.1 Dataset 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 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. Then the dataset is split into training and testing set with 30% for testing and 70% for training. Both the training and testing data are saved to disk.



```
dataset.arff
@data
36 -1,1,1,1,1,-1,-1,-1,-1,-1,1,1,1,-1,1,-1,-1,-1,-1,0,1,1,1,1,-1,-1,-1,-1,1,1,-1,-1
37 1,1,1,1,1,-1,0,1,-1,1,1,-1,1,0,-1,-1,1,1,0,1,1,1,1,-1,-1,0,-1,1,1,1,-1
38 1,0,1,1,1,-1,1,-1,-1,1,1,1,-1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,1,-1,1,0,-1,-1
39 1,0,1,1,1,-1,-1,-1,1,1,1,-1,-1,0,0,-1,1,1,0,1,1,1,1,1,-1,-1,1,-1,1,1,-1
40 1,0,1,1,1,-1,-1,-1,1,1,1,-1,-1,0,0,-1,1,1,0,1,1,1,1,1,-1,-1,1,-1,1,1,-1
41 1,0,-1,1,1,-1,1,1,1,-1,1,1,1,1,0,0,-1,1,1,0,-1,1,-1,-1,-1,0,-1,1,1,1,1
42 -1,0,-1,1,1,-1,1,1,1,-1,1,1,1,-1,1,0,0,-1,1,1,0,1,1,1,1,1,-1,-1,-1,1
43 1,0,-1,1,1,-1,-1,-1,1,1,1,1,-1,-1,0,-1,-1,-1,0,1,1,1,1,1,1,-1,-1,1,0,-1,-1
44 1,0,1,1,1,-1,-1,-1,1,1,1,-1,-1,0,-1,-1,1,1,0,1,1,1,1,-1,-1,0,-1,1,0,1,-1
45 1,0,-1,1,1,-1,1,1,1,-1,1,1,1,-1,1,0,1,-1,1,1,0,1,1,1,1,-1,1,1,0,1,1
46 1,1,-1,1,1,-1,-1,-1,1,1,1,1,0,0,1,-1,1,0,1,1,1,1,1,-1,0,-1,1,0,1,-1
47 1,1,1,1,1,-1,0,1,1,1,1,1,-1,0,0,-1,-1,-1,0,1,1,1,1,-1,1,1,1,1,-1,-1,1
48 1,1,-1,1,1,-1,1,1,-1,-1,1,1,1,1,-1,-1,-1,-1,-1,0,1,1,1,1,-1,-1,-1,-1,0,-1,-1
49 -1,1,-1,1,1,-1,-1,0,0,1,1,1,-1,-1,1,1,-1,1,1,0,-1,1,-1,1,1,-1,-1,-1,0,0,-1,-1
50 1,1,-1,1,1,-1,0,-1,1,1,1,1,1,-1,-1,-1,-1,1,0,1,1,1,1,-1,-1,-1,0,-1,1,1,-1
51 1,1,-1,1,1,1,-1,1,-1,1,1,1,-1,1,0,1,1,1,0,1,1,1,1,1,1,-1,1,-1,1,1,1
52 1,-1,-1,-1,1,-1,0,0,1,1,1,1,1,-1,-1,0,-1,1,1,0,1,1,1,1,1,-1,-1,-1,0,1,-1
53 1,-1,-1,1,1,-1,1,1,-1,1,1,1,-1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,-1,0,-1,1,1,-1,-1
54 1,-1,1,1,1,-1,-1,0,1,1,-1,1,1,1,0,-1,-1,-1,-1,0,1,1,1,1,-1,1,1,1,-1,1,-1
55 1,1,1,1,1,-1,-1,1,1,1,1,-1,-1,0,-1,-1,-1,-1,0,1,1,1,1,-1,-1,1,1,-1,1,1
56 1,1,1,1,1,-1,-1,1,-1,1,1,1,1,0,0,-1,-1,-1,0,-1,-1,1,-1,-1,-1,0,-1,1,0,-1,1
57 1,0,-1,1,1,1,-1,0,1,-1,1,1,1,1,0,0,-1,-1,-1,0,-1,1,-1,1,1,1,1,-1,-1,-1,1
58 1,0,1,1,1,-1,0,1,1,1,1,-1,-1,0,-1,-1,-1,-1,-1,0,1,1,1,1,-1,1,-1,1,0,-1,1
59 1,1,1,1,1,-1,1,1,-1,-1,1,1,1,-1,1,0,0,-1,1,1,0,1,1,1,1,1,1,1,-1,1,1
60 1,1,1,1,1,-1,1,0,-1,1,1,1,1,0,0,-1,1,1,0,1,1,1,1,1,1,1,1,-1,-1,1,1
61 1,-1,-1,-1,1,-1,1,1,-1,1,1,1,-1,-1,0,0,-1,1,1,0,1,1,1,1,1,1,-1,-1,1,0,1,1
62 1,-1,1,1,1,-1,0,1,-1,1,1,1,1,1,0,-1,1,1,0,1,1,1,-1,1,1,-1,1,1,0,1,1
63 1,-1,1,1,1,-1,0,-1,1,1,1,-1,-1,-1,-1,-1,-1,-1,0,1,1,1,1,1,1,0,-1,1,-1,-1,-1
64 1,-1,-1,1,1,1,-1,1,1,1,1,1,-1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,-1,0,-1,1,0,-1,1
```



```

1  relation phishing
2
3  @attribute having_IP_Address { -1,1 }
4  @attribute URL_Length { 1,0,-1 }
5  @attribute Shortining_Service { 1,-1 }
6  @attribute having_At_Symbol { 1,-1 }
7  @attribute double_slash_redirecting { -1,1 }
8  @attribute Prefix_Suffix { -1,1 }
9  @attribute having_Sub_Domain { -1,0,1 }
10 @attribute SSLfinal_State { -1,1,0 }
11 @attribute Domain_registration_length { -1,1 }
12 @attribute Favicon { 1,-1 }
13 @attribute port { 1,-1 }
14 @attribute HTTPS_token { -1,1 }
15 @attribute Request_URL { 1,-1 }
16 @attribute URL_of_Anchor { -1,0,1 }
17 @attribute Links_in_tags { 1,-1,0 }
18 @attribute SFH { -1,1,0 }
19 @attribute Submitting_to_email { -1,1 }
20 @attribute Abnormal_URL { -1,1 }
21 @attribute Redirect { 0,1 }
22 @attribute on_mouseover { 1,-1 }
23 @attribute RightClick { 1,-1 }
24 @attribute popUpWidnow { 1,-1 }
25 @attribute Iframe { 1,-1 }
26 @attribute age_of_domain { -1,1 }
27 @attribute DNSRecord { -1,1 }
28 @attribute web_traffic { -1,0,1 }
29 @attribute Page_Rank { -1,1 }

```

4.1.2 Training

The training data is loaded from the disk by the preprocessing module. A random forest classifier is then trained on the data using the scikit-learn library. Random Forest, being an ensemble learning technique, utilizes a collection of decision tree estimators. In this case, an ensemble of 10 decision tree estimators is used. Each decision tree follows the CART algorithm and aims to reduce the Gini impurity. Furthermore, the cross-validation score is calculated on the training data, while the F1 score is computed on the testing data. Subsequently, the trained model is exported to JSON using the next module.

4.1.3 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 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.

4.1.4 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

```
{
  "n_features": 17,
  "n_classes": 2,
  "classes": [-1, 1],
  "n_outputs": 1,
  "n_estimators": 10,
  "estimators": [{
    "type": "split",
    "threshold": "<float>",
    "left": {},
    "right": {}
  },
  {
    "type": "leaf",
    "value": ["<float>", "<float>"]
  }
]
```

Figure 4.4 Random Forest JSON structure

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

4.1.5 Exporting Algorithm

The algorithm used to export Random Forest model as JSON is as follows.

TREE_TO_JSON(NODE):

1. $\text{tree_json} \leftarrow \{\}$
2. **if** (node has threshold) **then**
3. $\text{tree_json}["\text{type}"] \leftarrow \text{"split"}$
4. $\text{tree_json}["\text{threshold}"] \leftarrow \text{node.threshold}$
5. $\text{tree_json}["\text{left}"] \leftarrow \text{TREE_TO_JSON}(\text{node.left})$
6. $\text{tree_json}["\text{right}"] \leftarrow \text{TREE_TO_JSON}(\text{node.right})$
7. **else**
8. $\text{tree_json}["\text{type}"] \leftarrow \text{"leaf"}$
9. $\text{tree_json}["\text{values}"] \leftarrow \text{node.values}$
10. **return** tree_json

RANDOM_FOREST_TO_JSON(RF):

1. $\text{forest_json} \leftarrow \{\}$
2. $\text{forest_json}['\text{n_features}'] \leftarrow \text{rf.n_features_}$
3. $\text{forest_json}['\text{n_classes}'] \leftarrow \text{rf.n_classes_}$
4. $\text{forest_json}['\text{classes}'] \leftarrow \text{rf.classes_}$
5. $\text{forest_json}['\text{n_outputs}'] \leftarrow \text{rf.n_outputs_}$
6. $\text{forest_json}['\text{n_estimators}'] \leftarrow \text{rf.n_estimators}$
7. $\text{forest_json}['\text{estimators}'] \leftarrow []$
8. $e \leftarrow \text{rf.estimators}$
9. **for** ($i \leftarrow 0$ **to** rf.n_estimators)
10. $\text{forest_json}['\text{estimators}'][i] \leftarrow \text{TREE_TO_JSON}(e[i])$
11. **return** forest_json

Chapter 5

Results

The output obtained in various stages :

5.1 Training

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

```
~/D/m/phishing_detector ➤ *+ ➤ backend/classifier ➤ python3 tr
/usr/local/lib/python3.7/site-packages/sklearn/ensemble/weight_bo
s is an internal NumPy module and should not be imported. It will
from numpy.core.umath_tests import inner1d
X_train:(7738, 17), y_train:(7738,)
Cross Validation Score: 0.9455923597113163
Accuracy: 0.9469400060295448
```

5.2 Plugin Feature Extraction

The 17 features extracted for the webpage at thetechcache.science are logged in to the console which is shown in figure 6.4. The features are stored as key value pairs and the values are encoded from -1 to 1 as discussed above.

```
▼ Object 1
  (-) Prefix/Suffix in domain: "-1"
  @ Symbol: "-1"
  Anchor: "-1"
  Favicon: "-1"
  HTTPS: "-1"
  HTTPS in URL's domain part: "-1"
  IP Address: "-1"
  No. of Sub Domains: "-1"
  Port: "-1"
  Redirecting using //: "-1"
  Request URL: "0"
  SFH: "-1"
  Script & Link: "0"
  Tiny URL: "-1"
  URL Length: "-1"
  iFrames: "-1"
  mailto: "-1"
```

5.3 Classification

The output of the classification is shown right in the Plugin UI. Green circle indicates legitimate site and Light red indicates phishing.

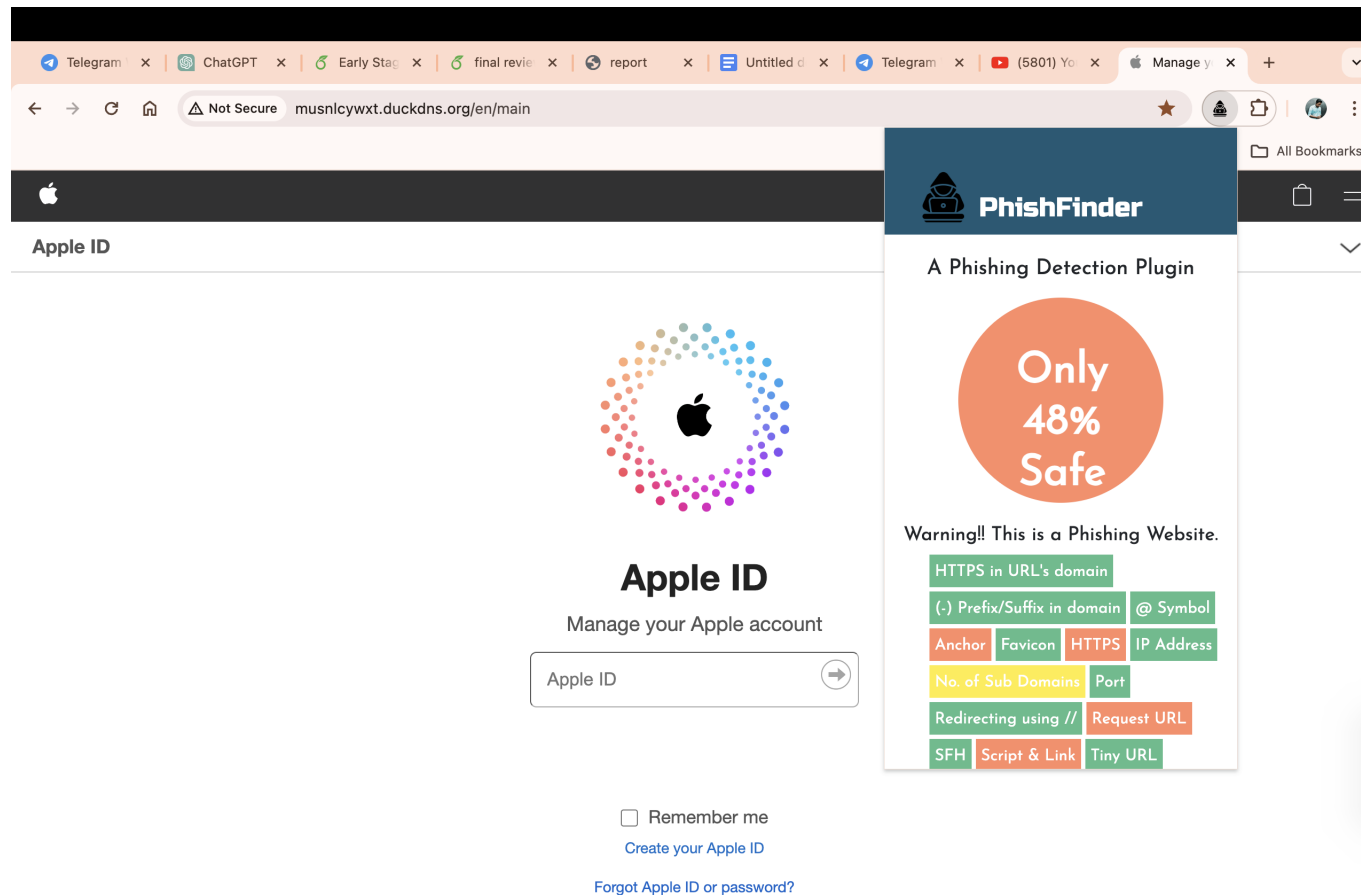


Figure 5.1: Phishing Website

5.4 Social Relevance

The project's social relevance lies in its ability to address a pressing concern in today's digital society: the proliferation of phishing websites and the resultant threats to users' online security. Phishing attacks have become increasingly sophisticated, often targeting individuals through deceptive emails, messages, or websites with the aim of stealing personal information, financial data, or login credentials. Such attacks can have devastating consequences, including identity theft, financial loss, and unauthorized access to sensitive accounts.

In this context, the proposed browser extension plays a crucial role in empowering users to defend themselves against phishing threats. By leveraging machine learning techniques to analyze website URLs in real-time, the extension provides users with immediate alerts when potential phishing websites are detected. This proactive approach mitigates the risk of users unknowingly falling victim to phishing scams, thereby safeguarding their personal and financial well-being.

Furthermore, the extension's capability to provide detailed information about the safety level of websites enhances users' awareness and understanding of online security risks. By clicking on the extension icon, users can access insights into the features contributing to a website's classification as phishing or safe. This transparency fosters a culture of informed decision-making, enabling users to assess the credibility of websites before interacting with them.

The social relevance of the project extends beyond individual users to encompass broader societal implications. As more individuals adopt the browser extension and benefit from its protective features, the collective resilience against phishing attacks increases. By reducing the success rate of phishing attempts, the project contributes to the overall reduction of cybercrime and its associated societal costs, including financial losses, reputational damage, and psychological harm.

Moreover, the project aligns with broader initiatives aimed at promoting digital literacy and cybersecurity awareness among internet users. By raising awareness about phishing threats and providing practical tools for defense, the project empowers individuals to navigate the digital landscape with confidence and resilience. This, in turn, contributes to the creation of a safer and more secure online environment for all users, fostering trust and confidence in digital interactions.

In summary, the social relevance of the project lies in its capacity to protect individuals from phishing attacks, enhance cybersecurity awareness, and contribute to the collective effort to combat cybercrime. By empowering users with proactive detection mechanisms and informative alerts, the project strengthens the resilience of individuals and society against online threats, ultimately fostering a safer and more secure digital ecosystem for all.

5.5 Future Scope

- **Enhanced Detection Capabilities:**
Further refinement and optimization of the machine learning model to improve the accuracy and efficiency of phishing detection. Integration of advanced algorithms and techniques, such as deep learning or natural language processing, to analyze website content and behavior for more sophisticated phishing detection.
- **Multi-Platform Support:**
Extension of the project to support additional web browsers beyond the initial target platforms, ensuring compatibility and accessibility for a wider user base. Development of mobile versions of the browser extension to extend protection to users across various devices and operating systems.
- **Community-driven Threat Intelligence:**
Implementation of mechanisms for crowdsourced threat intelligence, allowing users to report suspected phishing websites and contribute to the collective defense against emerging threats. Integration with threat intelligence feeds and cybersecurity research databases to enrich the detection capabilities with real-time insights into evolving phishing tactics and trends.

- **Advanced User Interaction:**
Expansion of interactive features within the extension, such as real-time phishing risk assessments for links embedded in emails, social media posts, or instant messages. Integration with password managers and security suites to provide seamless access to security-related functionalities and streamline users' cybersecurity workflows.

Chapter 6

Conclusion

The PhishFinder project is a groundbreaking initiative in online security, utilizing advanced 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 dynamic nature of the digital landscape. The project emphasizes speed and accuracy in phishing threat detection. 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 sophistication of phishing attacks. The project's iterative development, informed by user feedback, highlights its adaptability and commitment to continuous improvement. Privacy and transparency measures have been implemented to ensure user data security, enhancing the system's overall reliability and trustworthiness. PhishFinder not only represents a technological advancement but also serves as a proactive response to the escalating 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.

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.

Appendices

Appendix A

CODE

A.0.1 Machine Learning Model

- Importing Necessary Libraries

```
import arff

import numpy as np

import json

from sklearn.model_selection import

    train_test_split , KFold
```

- Data Preprocessing

```
dataset = arff.load(open('dataset.arff', 'r'))

    )

data = np.array(dataset['data'])

X, y = data[:, :-1], data[:, -1]

y.reshape(y.shape[0])

print('Before splitting ')

print('X:{0}, y:{1}'.format(X.shape, y.shape))

X_train, X_test, y_train, y_test = train_test_split

    (X, y, test_size=0.3, random_state=0)

print('After splitting ')

print('X_train:{0}, y_train:{1}, X_test:{2}, y_test

    :{3}'.format(X_train.shape, y_train.shape,

X_test.shape, y_test.shape))
```

– Training Process

```
X_train = np.load('../dataset/X_train.npy')
y_train = np.load('../dataset/y_train.npy')
print('X_train:{0}, y_train:{1}'.format(X_train.
    shape, y_train.shape))
```

– Accuracy

```
X_test = np.load('../dataset/X_test.npy')
y_test = np.load('../dataset/y_test.npy')
pred = clf.predict(X_test)
print('Accuracy: {}'.format(accuracy_score(y_test,
    pred)))
```

– Performance Measure

```
clf = RandomForestClassifier()
print('Cross Validation Score: {}'.format(np.mean(
    cross_val_score(clf, X_train, y_train, cv=10))))
```

Appendix B

Screenshots

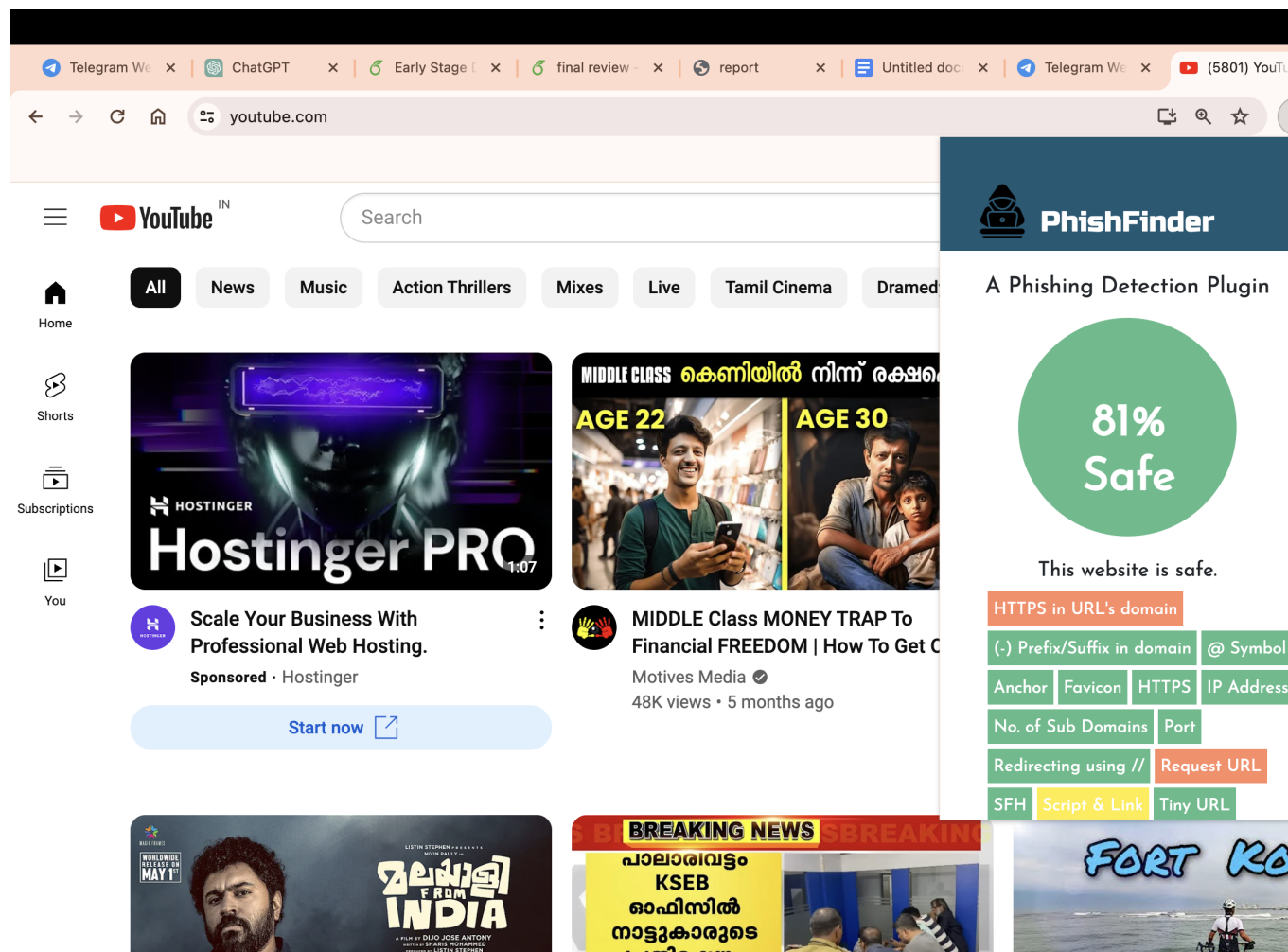


Figure B.1: Legitimate Website

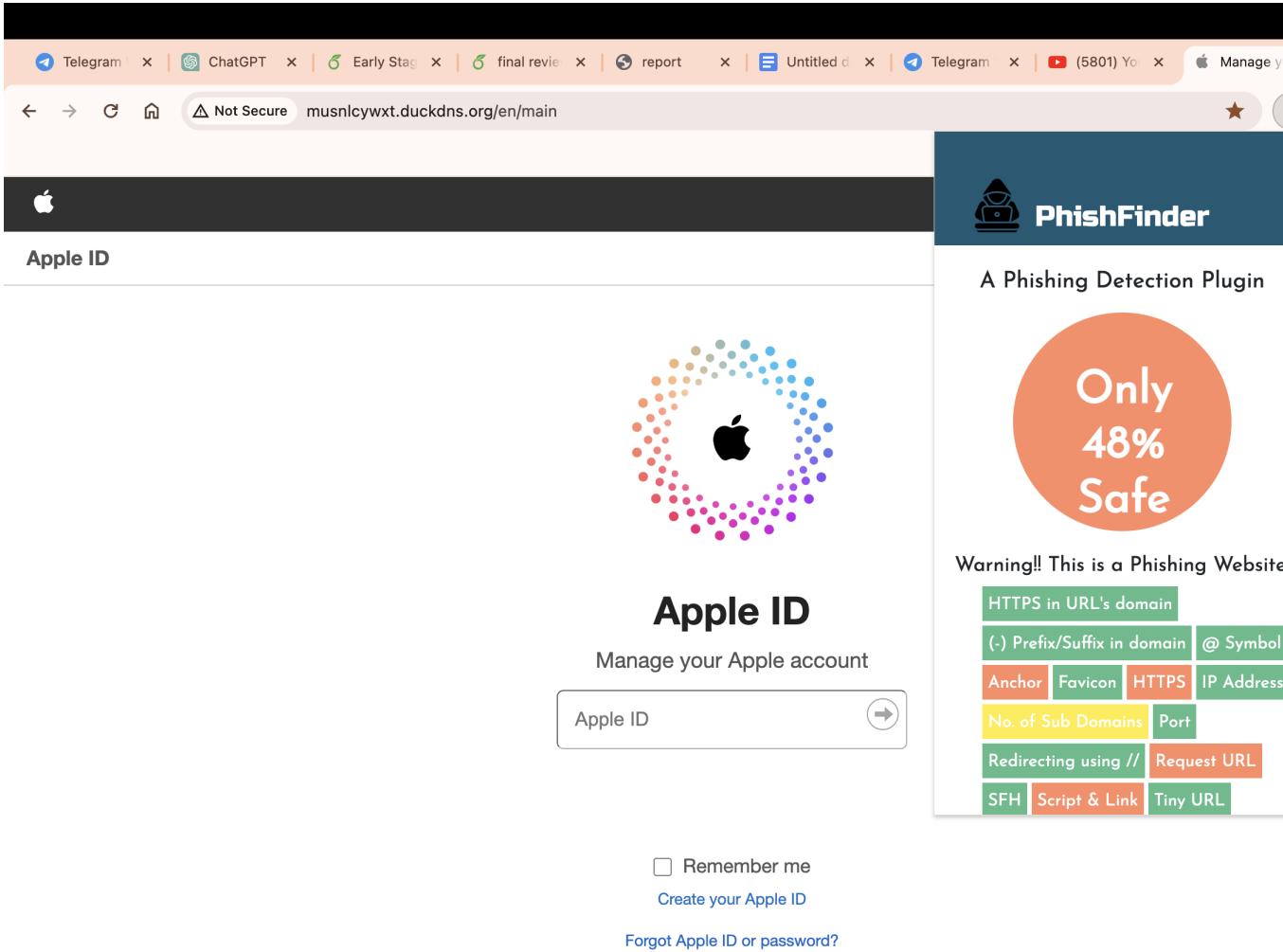


Figure B.2: Phishing Website

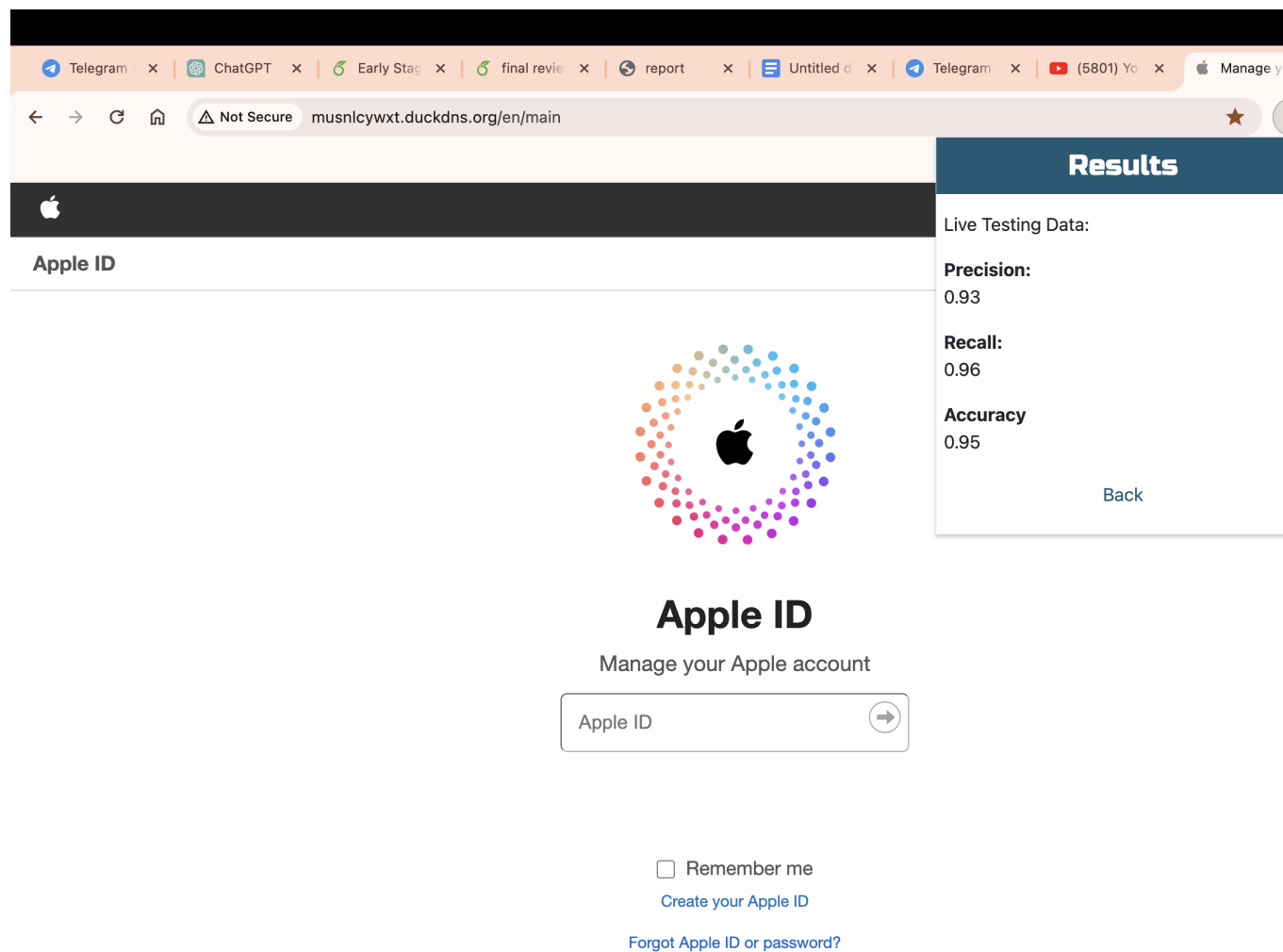


Figure B.3: Parameters

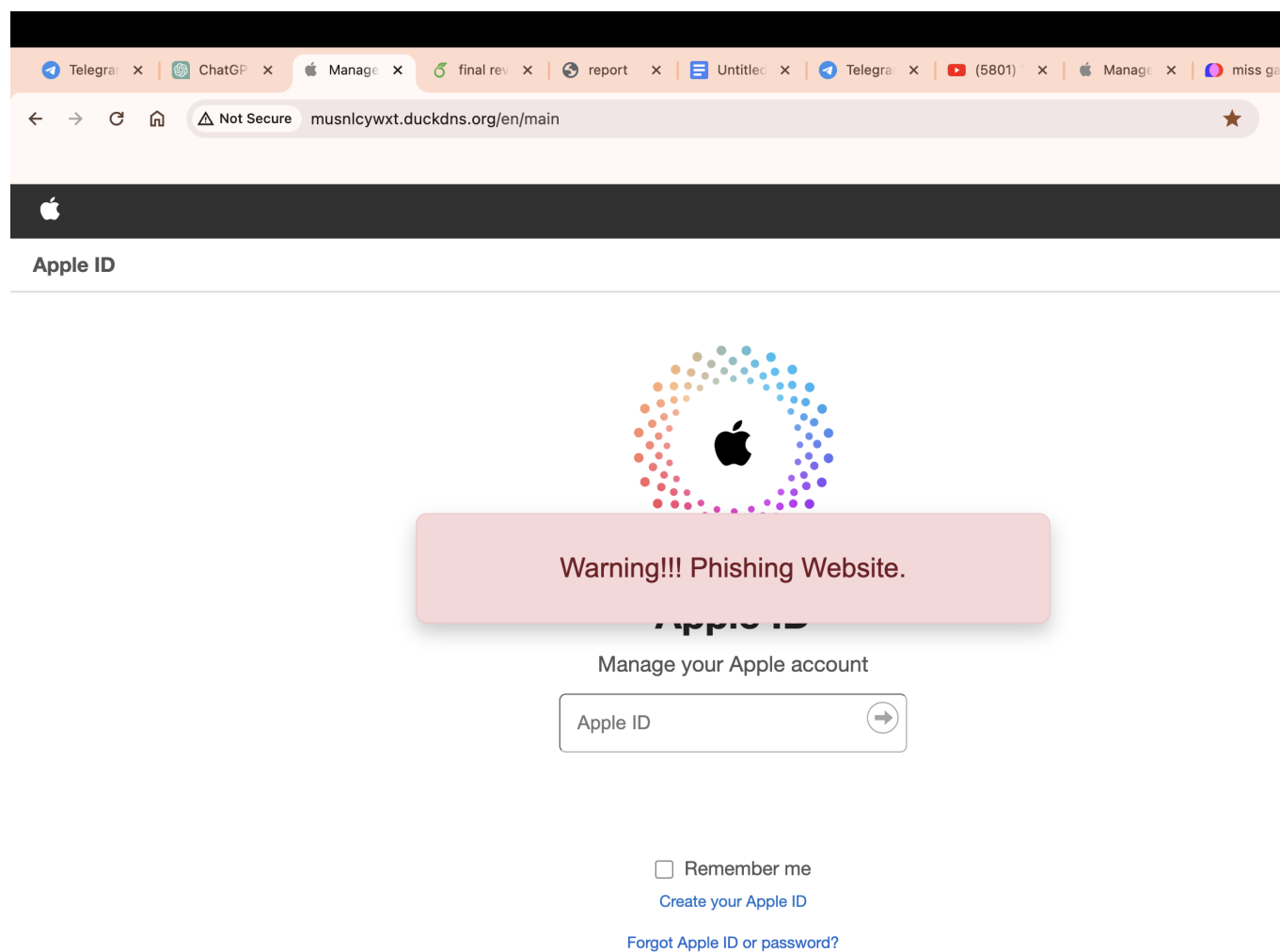


Figure B.4: Warning