Report on

Intelligent System for Identification of Malicious Websites

Submitted in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

By

Mr. Ajay Shitkar (Roll No. 14)

Mr. Prathmesh Malvi (Roll No. 38)

Mr. Yash Davare (Roll No. 53)

Under the Guidance of

Dr. Dinesh Patil

Department of Computer Engineering

Vidyavardhini's College of Engineering & Technology, Vasai (W)

(NAAC and NBA Accredited)

(Affiliated to University of Mumbai)

(2023-2024)

CERTIFICATE

This is to certify that the project entitled **“Intelligent System for Identification of Malicious Websites”** is a bonafide work of **“Ajay Shitkar (Roll No. 14), Prathmesh Malvi (Roll No. 38) and Yash Davare (Roll No. 53)”** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of “**Bachelor of Engineering**” in “**Computer Engineering**”.

Dr. Dinesh Patil
Guide

Dr. Megha Trivedi
Head of Department

Dr. Harish Vankudre
Principal

Project Report Approval for B.E.

This project report entitled **"Intelligent System for Identification of Malicious Websites"** by **'Ajay Shitkar (Roll No. 14), Prathmesh Malvi (Roll No. 38) and Yash Davare (Roll No. 53)'** is approved for the degree of '**Bachelor of Engineering**' in '**Computer Engineering**'.

Examiners

1. ____________________________

2. ____________________________

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Ajay Shitkar (14)

Prathmesh Malvi (38)

Yash Davare (53)

Date:

Acknowledgements

We would like to extend our heartfelt gratitude to several individuals who played pivotal roles in the successful completion of our project. First and foremost, we express our deepest appreciation to Dr. Dinesh Patil, who served as our mentor and our HOD Dr. Megha Trivedi their guidance, expertise, and unwavering support were instrumental throughout our research journey. We also acknowledge the invaluable contributions of our esteemed Principal, Dr. H.V. Vankudre, for their encouragement and belief in our capabilities. Furthermore, we cannot overlook the significant influence of our friends and family, who provided both direct and indirect assistance, serving as pillars of strength during challenging times. Additionally, we are indebted to the numerous academicians who generously served as supervisors and assessors, offering their expertise and insights that greatly enriched our project. Their collective efforts have been indispensable, and we express our sincere gratitude to each of them for making this study a resounding success.

Ajay Shitkar (14)

Prathmesh Malvi (38)

Yash Davare (53)

Date:

ABSTRACT

Phishing is defined as mimicking a creditable company's website aiming to take private information of a user. In order to eliminate phishing, different solutions are proposed. Data mining is a promising technique used to detect phishing attacks. In this implementation, an intelligent system to detect phishing attacks is developed. We used to feature mining technique to decide categories of websites: legitimate or phishing. Different classifiers were used in order to construct accurate intelligent system for phishing website detection. Classification accuracy, area under receiver operating characteristic. Results showed that Random Forest has outperformed best among the classification methods by achieving the highest accuracy. Random forest runtimes are quite fast, and it can deal with different websites for phishing detection

We are implementing a chrome extension to detect the Phishing websites and alert the user using machine learning and deep learning techniques whenever a user visits the website. The project aims to gather URLs from multiple sources, including Kaggle, Phish Tank, and Alexa URL. Whenever a user opens a website on his/her browser, the extension will work in the background. If the website is detected as phishing website, the user will be alerted with a pop-up box with an 'OK' button. If the website is not detected as phishing website, then no action will be taken. We have used various machine learning algorithm Random Forest

Index

List Of Figures

Chapter 1

INTRODUCTION

1.1 Introduction

Different types of web threats may yield identity theft, stealing of private information, financial loss and loss of customer's assurance in online banking and e-commerce. Hence, internet suitability for commercial transactions becomes hesitant. Phishing is a form of web threats which is defined as impersonating a website of a creditable company. Fraudulent persons create phishing websites to mimic web pages of genuine websites. The honest internet users are defrauded by these phishing websites which are similar to the legitimate ones. Since the number of phishing websites are increasing day by day, it will become a serious problem, even for the experienced users in the computer security and internet.

In order to distinguish honest and phishing websites, two different approaches are used in recognizing phishing websites. The first one checks if the requested URL is on the blacklists by comparing with those in that list [1]. Meta-heuristic methods in which quite a lot features are collected from the website to categorize it as either legitimate or phishing website is the second approach [2]. The accurateness of the meta-heuristic method is based on extracting a set of distinguishing features which may help in differentiating the website. CNN Algorithm is generally used to extract the features from the websites to find patterns as well as relationships between them. Data mining algorithms are highly imperative for decision-making, since decisions can be made based on the rules accomplished from a data-mining algorithm.

The mini project's contribution is to provide an accurate and reliable Phishing Detection System. By implementing various machine learning algorithm's we have implemented the system that would to detect phishing URL's. The classification can be done using various features of URL. Overall, the mini project's contribution is to provide a practical and reliable solution for protecting the confidential and sensitive information of users.

1.2 Problem Statement & Objectives

Problem Statement:

In today's world, with increasing use of the internet and digital platforms, the risk of online attacks such as phishing has also increased significantly. Phishing is a form of online attack where cybercriminals create fake websites or emails that mimic legitimate ones to deceive users into providing sensitive information. This confidential data has the potential to be exploited for illicit activities such as financial fraud or identity theft.

Objective:

- Develop a Chrome browser extension that can detect phishing websites in real time as users browse the web.

- Ensure user privacy by avoiding the need to send URLs to external servers for classification, thereby preventing the exposure of sensitive information.

- Provide instant warnings to users when they access potential phishing websites, reducing the risk of falling victim to phishing attacks due to network latency.

- Create a user-friendly user interface within the Chrome extension, offering clear and intuitive warnings or notifications when a user encounters a suspicious website.

1.3 Scope

The scope of Phishing Detection System is to form the legitimate and phishing URL's, so that we will able to protect confidential and sensitive data of user. This process include the using of various algorithms such as CNN [3] and Random Forest Algorithm [4]. The extension can made to cache results of frequently visited sites and hence reducing computation. Phishing website detection can greatly benefit from advances in multi-modal analysis, allowing the system to effectively identify threats in multimedia content and user interactions.

Chapter 2

LITERATURE REVIEW

2.1 Literature Survey

Machine learning-based systems have gained prominence in recent years. These systems use algorithms to analyze and classify emails, URLs, and other online content, detecting patterns and anomalies indicative of phishing. They adapt and improve their detection capabilities over time as they encounter new phishing tactics. Many systems employ URL filtering and reputation-based techniques to assess the legitimacy of sender domains and URLs. They compare these with known lists of malicious entities to block or flag suspicious emails and websites [1].

Moreover, some systems use heuristics and behavioral analysis to identify phishing attacks. They scrutinize the behavior of users, applications, and network traffic to uncover deviations from normal patterns, signaling potential phishing attempts. An existing chrome plugin named Phish-Detector uses a rule-based approach so that it can detect phishing without external web service. Although rule-based approaches support easier implementation on client side, they can't be accurate compared to Machine Learning based approaches.

Similar work on detection of phishing attacks are carried using genetic algorithm uses a rule that is generated by a genetic algorithm for detection. Phish Net is one such Predictive blacklisting approach. It used rules that can match with TLD, directory structure, IP address, HTTP header response and some other. There are five approaches to anti-phishing are Rules based, Blacklisting, Content based, Machine Learning based
and Hybrid [2].

<u>Random Forest Algorithm</u>

Training:

- **Random Sampling with Replacement:** Random Forest builds multiple decision trees by randomly selecting a subset of the training data.
- **Feature Selection:** At each node of the decision tree, a random subset of features (a subset of the total available features) is considered for splitting.
- **Tree Building:** Each tree is grown by recursively splitting the nodes. It looks for the best split among the randomly selected subset of features.

Prediction:

- For classification tasks, when making predictions, each tree in the forest "votes" for a class, and the most common class label is chosen as the final prediction.
- For regression tasks, the predictions from each tree are averaged to get the final prediction.

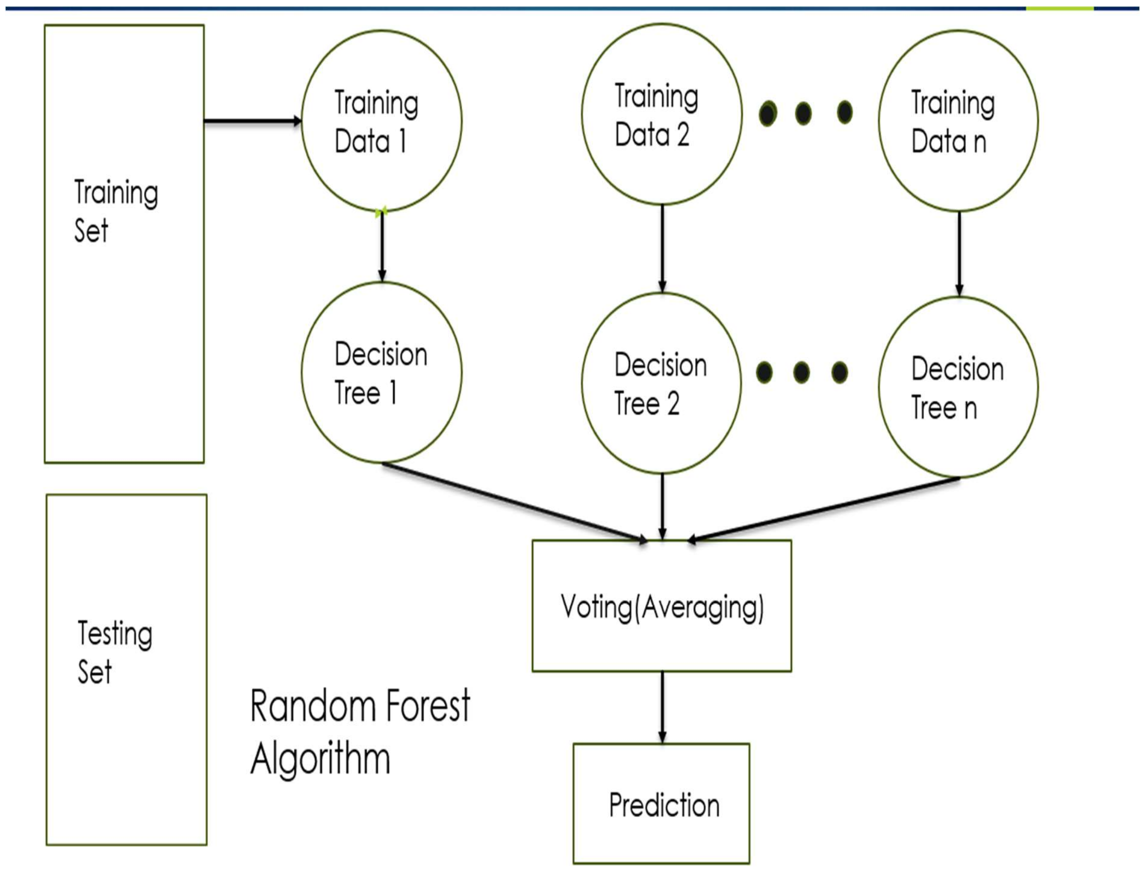


Fig.2.1 Random Forest Algorithm

Fig.2.1 represents random forest algorithm an ensemble learning method that combines multiple decision trees to make more accurate predictions. It works by training a collection of decision trees on different subsets of the data and averaging their predictions to improve overall performance, making it a robust and powerful machine learning technique.

Pseudo Code For Random Forest Algorithm

Input: A dataset with features (independent variables) and corresponding target labels (dependent variable)

Output: The algorithm produces predictions or classifications for new, unseen data points based on the trained random forest model.

Terminology: Decision Trees, Ensemble, Bootstrap Sampling.

Code:

```
Step 1:  Define a function Random Forest that takes two arguments, a dataset S and a set of features F.

Step 2:  Initialize an empty set H to store the ensemble of decision trees.

Step 3:  For each iteration i from 1 to B, where B is the number of trees to be created, do the following:

Step 4:  Create a bootstrap sample S(i) by randomly selecting data points with replacement from the dataset S.

Step 5:  Call a function RandomizedTreeLearn to build a randomized decision tree h_i using the bootstrap sample S(i) and a very small subset of features f (from the set F).

Step 6:  Add the learned decision tree h_i to the ensemble H.

Step 7:  End the loop.

Step 8:  Return the ensemble H containing all the decision trees.

Step 9:  End the Random Forest function.

Step 10: Define a function RandomizedTreeLearn that takes two arguments, a dataset S and a set of features F.

Step 11: At each node of the tree:

Step 12: Select a very small subset of features f from the set F.

Step 13: Split the current node on the best feature within the subset f.

Step 14: Return the learned decision tree.

Step 15: End the RandomizedTreeLearn function.
```

2.2 Limitations in Existing System

In existing systems, plug-ins use either a rule-based approach or a server-side ML-based approach. The rule-based approach does not seem to work well compared to ML-based approaches, and on the other hand, ML-based approaches need the support of libraries, so they were not implemented in the client-side extension. All existing plugins send the destination URL to an external web server for classification. This project aims to implement the same in a browser extension, removing the need for an external web service and improving user privacy.

Conclusion: The increasing prevalence of phishing attacks in the digital landscape has prompted the adoption of machine learning-based systems, such as the Random Forest Algorithm, to bolster email and URL security. While rule-based approaches exist, their accuracy is surpassed by machine learning methods. The need to balance security and user privacy, eliminating external web service reliance for more efficient and privacy-conscious protection.

Chapter 3
PROPOSED SYSTEM

3.1 System Architecture

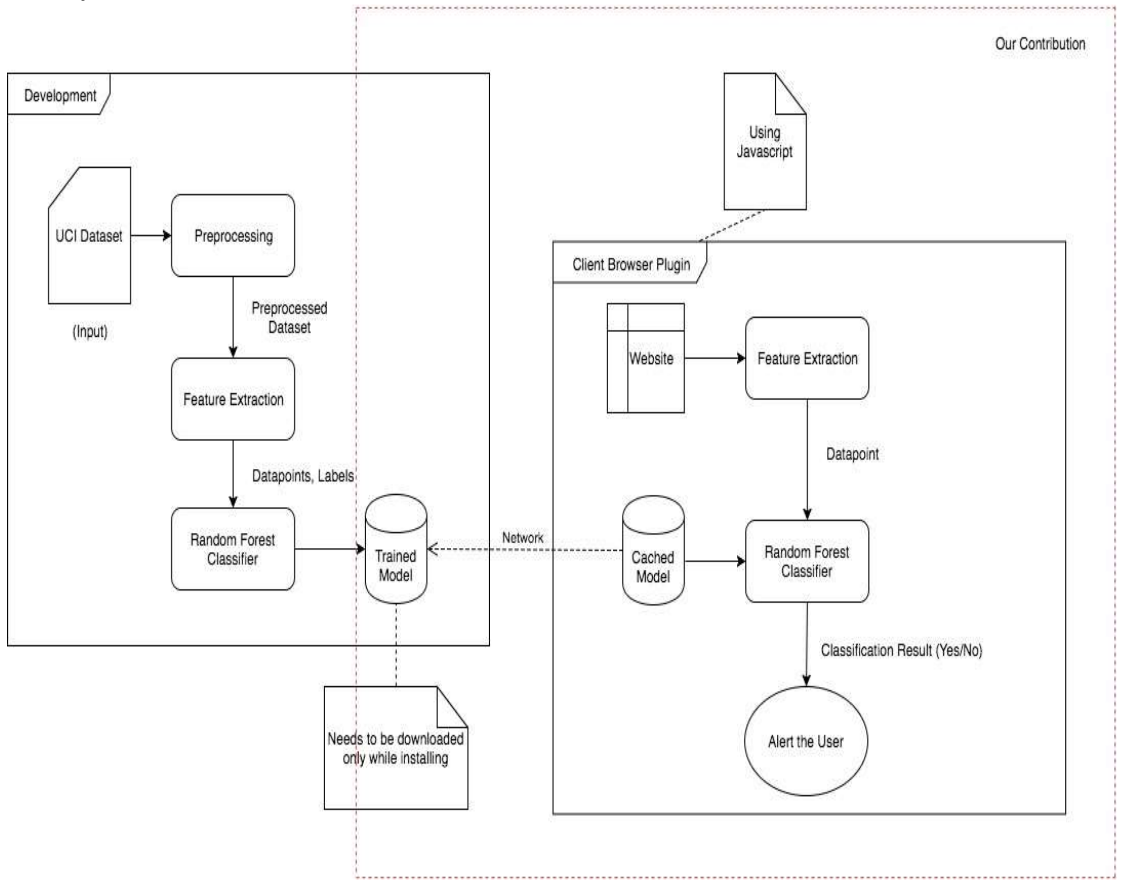


Fig. 3.1 Architecture

1. Fig.3.1 represents system architecture of our model. 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.

2. A browser script has been implemented which uses the exported model JSON to classify the website being loaded in the active browser tab. 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.

3. 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.

4. Once the features are encoded, the plugin then checks for the exported model JSON in cache and downloads it again in case 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, in case the website is classified as phishing.

3.2 Use Case Diagram of the System

The overall use case diagram of the entire system is shown in figure 3.2. The user can install the plugin and then can continue his normal browsing behavior. This plugin will automatically check the browsing pages for phishing and warns the user of the same.

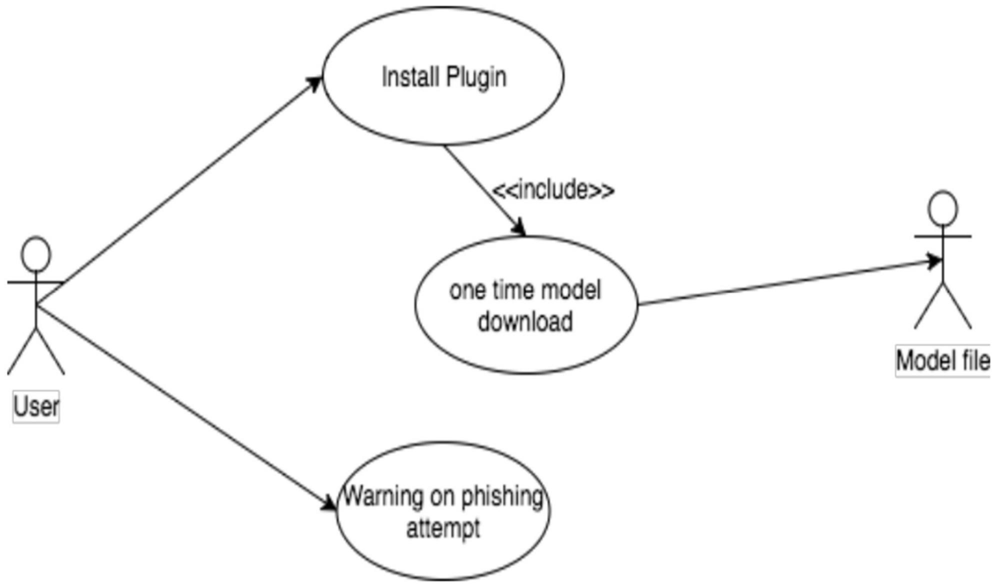


Fig. 3.2 Use case diagram

3.3 Algorithm

3.3.1 Random to JSON Exporting Algorithm

This algorithm used to export Random Forest model as JSON. It takes the learned Random Forest classifier object and the recursively generates its JSON representation

Terminology : tree_json , forest_json, n_features, n_estimators, n_outputs,

```
TREE_TO_JSON(NODE):
1. tree_json ← {}
2. if (node has threshold) then
3. tree_json[“type”] ← “split”
4. tree_json[“threshold”] ← node.threshold
5. tree_json[“left”] ← TREE_TO_JSON(node.left)
6. tree_json[“right”] ← TREE_TO_JSON(node.right)
7. else
8. tree_json[“type”] ← “leaf”
9. tree_json[“values”] ← node.values
10. return tree_json
```

```
RANDOM_FOREST_TO_JSON(RF):
1. forest_json ← {}
2. forest_json['n_features'] ← rf.n_features_
3. forest_json['n_classes'] ← rf.n_classes_
4. forest_json['classes'] ← rf.classes_
5. forest_json['n_outputs'] ← rf.n_outputs_
6. forest_json['n_estimators'] ← rf.n_estimators
7. forest_json['estimators'] ← []
8. e ← rf.estimators
9. for (i ← 0 to rf.n_estimators)
10. forest_json[‘estimators’][i] ← TREE_TO_JSON(e[i])
11. return forest_json
```

3.4 Methodology of Proposed System

In today's digital age, cybersecurity has become a paramount concern, particularly when it comes to identifying fraudulent websites. To tackle this issue effectively, a comprehensive approach is required. This includes data collection, pre-processing, model selection, training, and evaluation. In this context, we'll explore a robust methodology that incorporates all these steps and culminates in the implementation of the Random Forest Algorithm for identifying legitimate and fraudulent websites. This multifaceted approach ensures both accuracy and efficiency in safeguarding online users from potentially harmful websites.

1. **Data Collection**

   The initial step is to gather data. Data collection involves the collection of a large dataset of both legit and fraudulent websites.

2. **Data Pre-processing**

   Once the data has been collected, the subsequent step involves pre-processing it. Data pre-processing encompasses cleaning and transforming the data to make it ready for analysis. The pre-processing step entails removing duplicates, managing missing values, and transforming the data into a format that is appropriate for machine learning algorithms.

3. **Model Selection**

   Model selection involves choosing the algorithm that can directly classify legit and fraud websites. For this project, we trained the model using various algorithms. For each the accuracy, F1-score, precision, recall was calculated. The algorithm which provided the best values was Random Forest Algorithm [4].

4. **Model Training**

   After selecting the algorithm, the next step involves training the model. Model training involves feeding the model with the pulled features and their corresponding labels. The data to be trained from the preprocessing module is loaded from the disk. A random forest classifier is trained on the data using scikit-learn library.

5. **Exporting Model**

   This module takes the learned Random Forest classifier object and the recursively generates its JSON representation which is written to file in disk. In this basically the trained model is exported to implement extension.

6. **Model Evaluation**

   After training the model, the next step is to estimate its performance. To evaluate the effectiveness of the model, it is tested on a separate test set that has not been used in training. The performance of the model is then measured using various metrics such as sensitivity, accuracy, and recall.

Conclusion: In this Chapter we had discuss about our approach to identifying fraudulent websites through the implementation of the Random Forest Algorithm represents a comprehensive and effective strategy. We meticulously progress through data collection, pre-processing, model selection, and training, ultimately exporting the trained model. This method yields a robust solution, with thorough evaluations ensuring its performance, accuracy, and ability to protect users from potentially harmful online entities.

Chapter 4
EXPERIMENT SETUP

4.1. Performance Evaluation

The performance evaluation of any system, particularly one aimed at detecting and interpreting URLs, is paramount to its effectiveness and reliability. In this context, assessing the accuracy, cross-validation results, real-time processing capabilities, user adaptability, and privacy and security features is crucial. These metrics not only gauge the system's efficiency but also its ability to safeguard users in real-world scenarios. This evaluation process provides a comprehensive understanding of the system's strengths and areas for potential improvement, ensuring a robust and user-centric solution.

- **Accuracy:** Measure the system’s ability to correct recognize and interpret URLs. Use metrics like cross validation, precision, recall, and F1 score to assess the accuracy of phishing URLs. The accuracy obtained after training is about **94.75429 %**

- **Cross Validation:** K-fold cross-validation can be a valuable technique when evaluating the performance of a phishing URL detection extension. K-fold cross-validation helps address some common issues in machine learning, such as overfitting and the potential for dataset bias. The cross-validation score obtained after training dataset is **94.76603**

- **Real Time Processing:** Evaluate the system’s real time processing capabilities.

- **User Adaptability:** To assess how quickly users can adapt to the system’s requirements and whether the system is able to deliver the necessary services for what it has been design.

- **Privacy and Security:** Ensure that the system protects the privacy of users, especially in scenarios where sensitive or confidential information is captured.

4.2 Hardware and Software Requirements

4.2.1 Hardware Requirements

- Processor: Quad Core
- Hard Disk: 120 GB
- RAM: 2 GB

4.2.2 Software Requirements

- Tech: Python, HTML/CSS/JavaScript.
- Operating system: Windows
- Library : scikit-learn, NumPy, liac-arff
- Software tools: Chrome
- IDE: Jupyter Notebook. Visual Studio Code, Google Collab

Conclusion : In this particular module , a URL detection system's performance evaluation, which includes accuracy (94.75%) and cross-validation (94.77%), demonstrates its robustness. The emphasis on real-time processing, user adaptability, privacy, and security underline its potential. Hardware requirements comprise a Quad-Core processor and 2GB RAM, while software needs include Python and relevant libraries. This holistic assessment ensures a dependable and user-centric solution.

Chapter 5

IMPLEMENTATION PLAN

5.1 Timeline Chart for Term 1

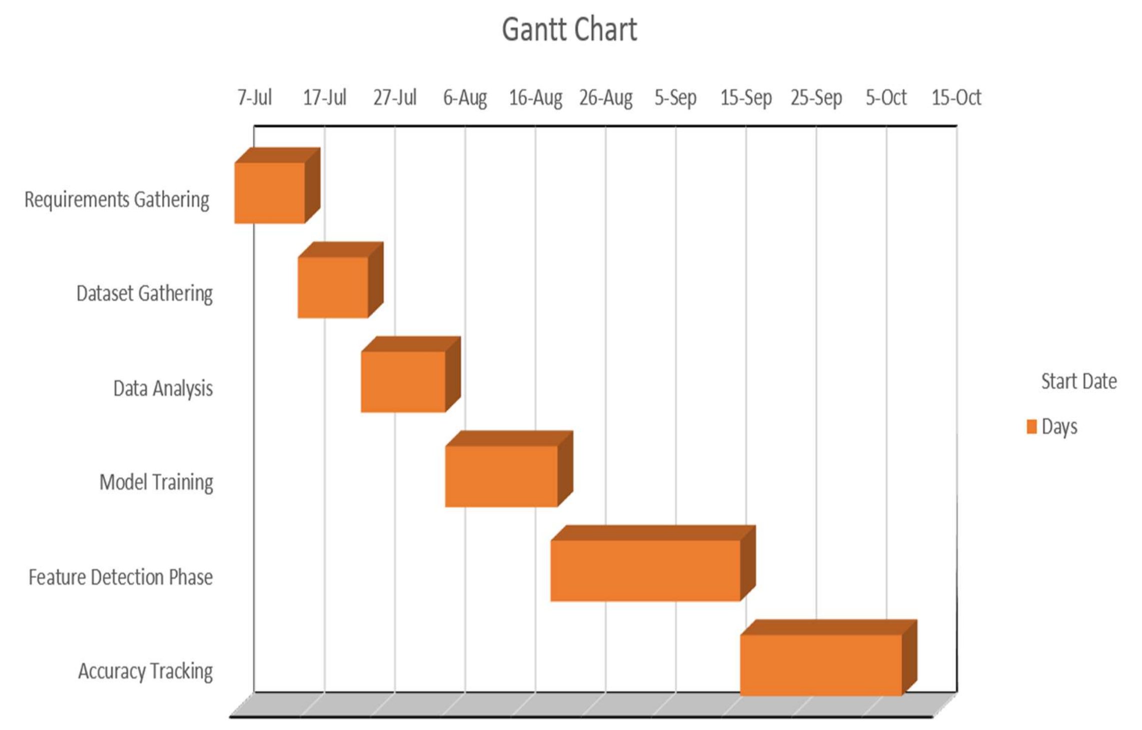


Fig. 5.1 Gantt Chart

Week 1 : Discuss about the project , problem domain, proposed system with guide.

Week 2 : Literature Survey is done followed by requirements gathering.

Week 3 : Gathered dataset suitable as per our requirements , testing it with algorithms

Week 4 : Selection of particular algorithm is done by testing under various condition.

Week 5: Model Training and data preprocessing is done.

Week 6 : Trained model exporting part is done and accuracy is calculated.

Week 7 : Finally worked on the frontend part .

Week 8 : Report , PPT and Handbook finalization is completed.

Chapter 6

RESULT

In this chapter , we will be discussing on various experimental results that we have obtained from our project and how user will use our system .

Output 1: Loading Extension into the browser by Load unpacked enabling developer mode

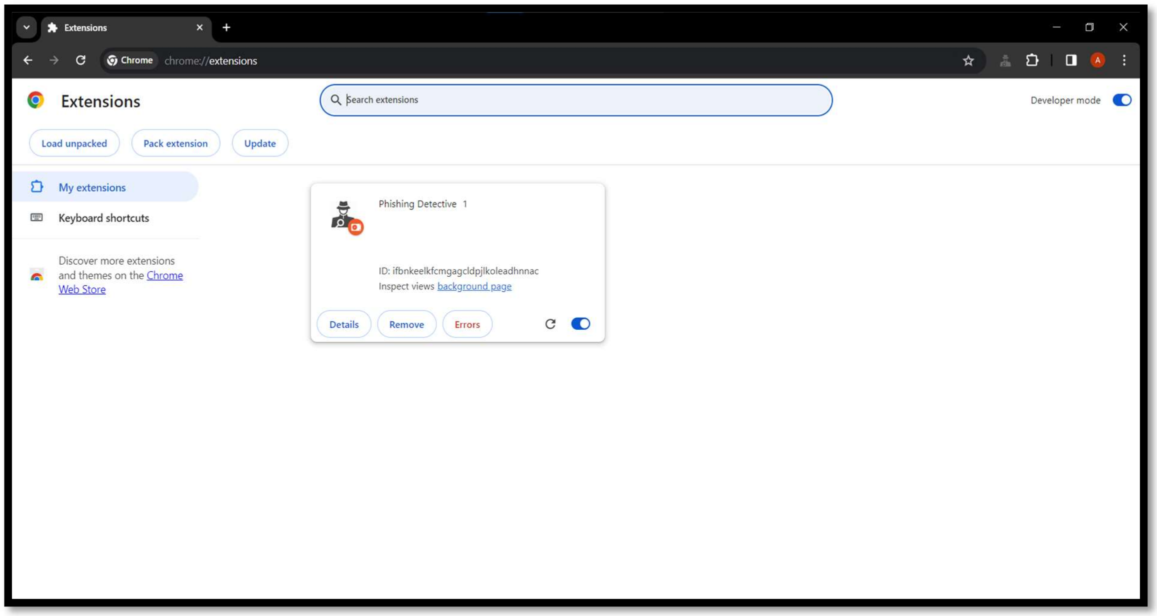


Figure 6.1.1 Installed Chrome Extension

Output 2: Loading URL Legitimate/ Phish into the URL address of our browser.

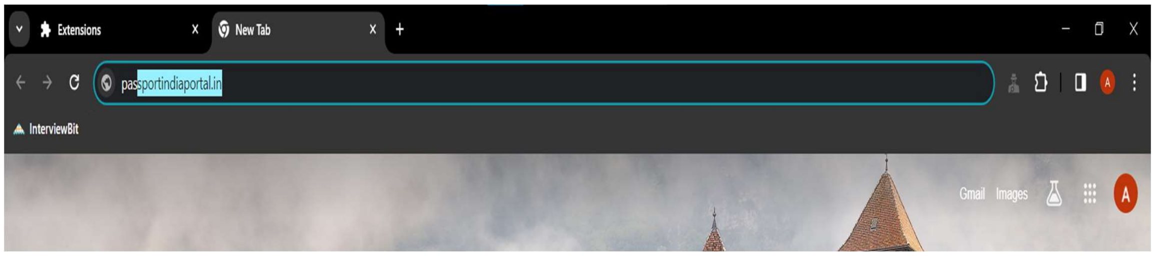
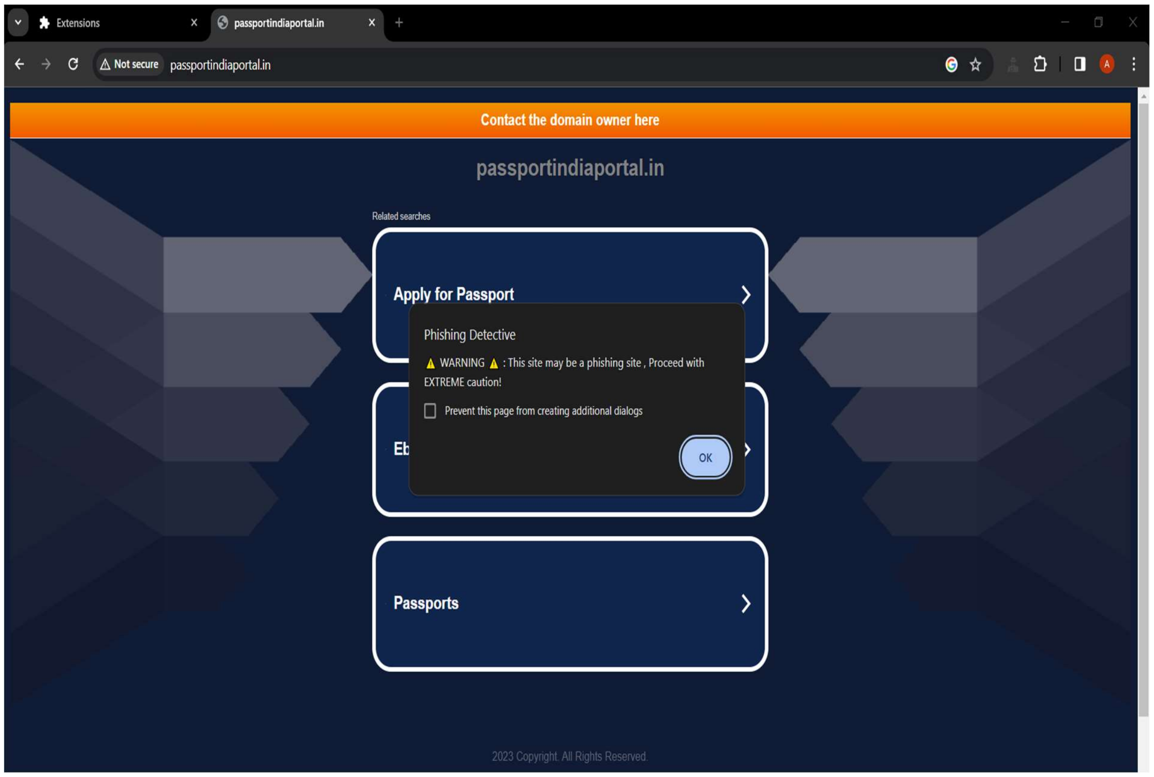


Figure 6.2.2 Input URL

Output 3: Once the phish/malicious website is detected it show alert pop-up window showing it is a phish site or a legitimate one.



Phishing Detective

⚠ WARNING ⚠ : This site may be a phishing site , Proceed with EXTREME caution!

☐ Prevent this page from creating additional dialogs

Figure 6.3.3 Phishing detected for malicious input URL

Chapter 7

CONCLUSION

It is a phishing detection system that focuses on client-side implementation with fast detection to warn users before phishing. The main application ports the Random Forest classifier to JavaScript. Similar works often use web page functions that cannot be extracted from the client side and this leads to network-dependent detection. On the other hand, this system only uses features that can be extracted from the client side and thus can provide fast detection and better privacy. Although the use of smaller functions leads to a slight decrease in accuracy, it increases the usability of the system. This work has identified a subset of web page functions that can be implemented on the client side without significantly affecting accuracy. A port from Python to JavaScript and Random Forest's own implementation in JavaScript further aided rapid identification, as the JSON representation of the model and the classification script were built with time complexity in mind. The plugin can detect phishing even before the page is fully loaded.

Future Work

The classifier is currently trained on 17 features which can be increased provided that, they don't make the detection slower or result in loss of privacy. The extension can made to cache results of frequently visited sites and hence reducing computation. Phishing website detection can greatly benefit from advances in multi-modal analysis, allowing the system to effectively identify threats in multimedia content and user interactions[5]. Extend the model to incorporate various data modalities, including images and user interactions, to enhance detection capabilities for phishing websites that use multimedia content and interactive elements. The future of this approach lies in its adaptability, resilience, and effectiveness in countering the ever-evolving tactics employed by phishers, ultimately safeguarding users from online threats.

In conclusion, this client-side phishing detection system, powered by a JavaScript-based Random Forest classifier, excels in rapid detection and enhanced privacy protection. Future enhancements may involve adding more features, optimizing caching for frequently visited sites, and exploring multi-modal analysis to combat evolving phishing tactics.

REFERENCES

1. *Intelligent phishing website detection using random forest classifier*. (2017, November 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8252051

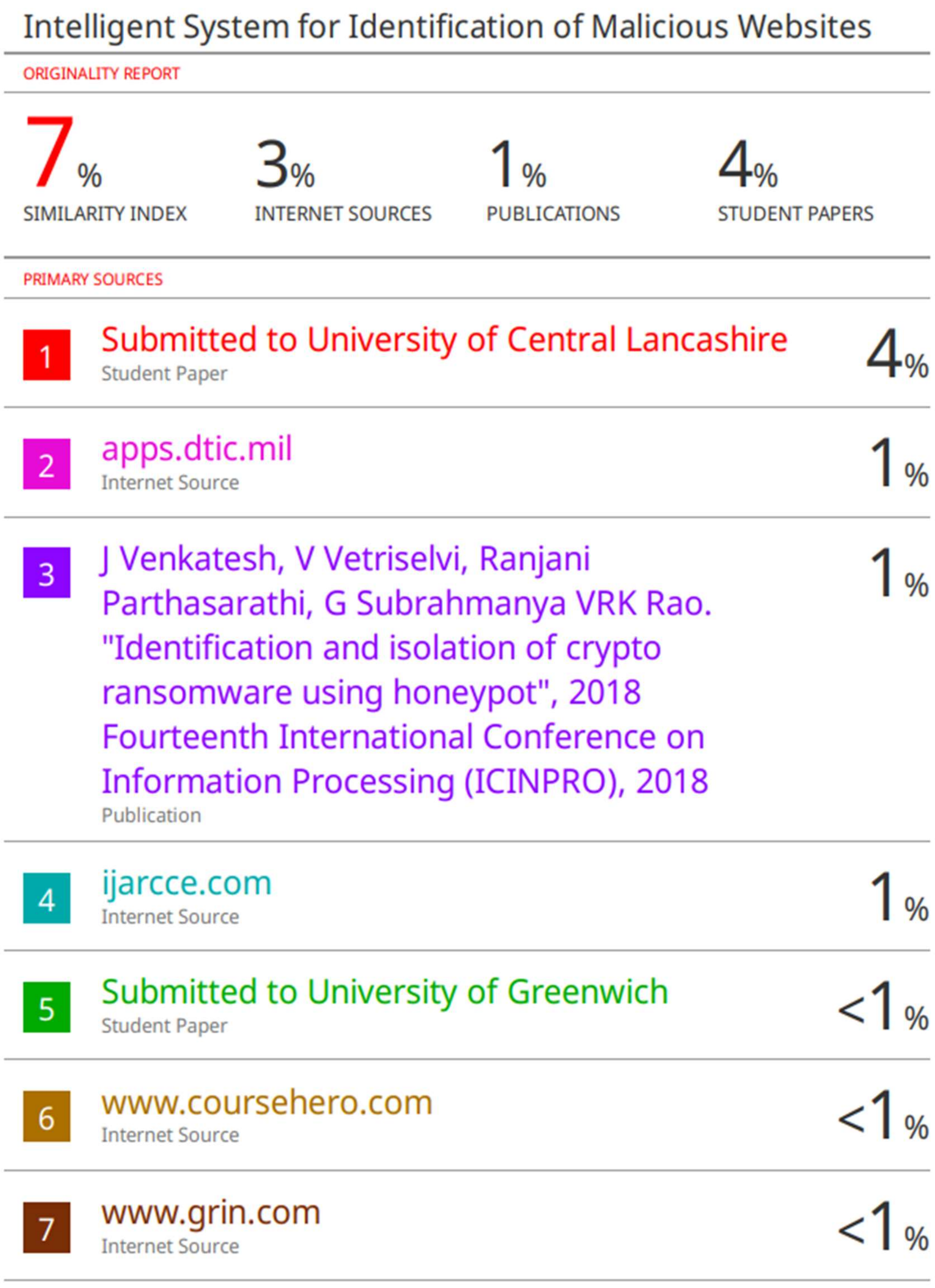
2. *Phishing Detection and Prevention using Chrome Extension*. (2022, June 6). IEEE Conference Publication | IEEE Xplore. https://ijcrt.org/papers/IJCRT2305779

3. *A Guide to Convolutional Neural Networks — the ELI5 way | Saturn Cloud Blog*. (2023, October 4). https://saturncloud.io/blog

4. R, S. E. (2023, October 26). *Understand Random Forest Algorithms With Examples (Updated 2023)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/

5. *VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity*.(October 2020). CCS '20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security | Research Gate : DOI:10.1145/3372297.3417233

PLAGARISM REPORT



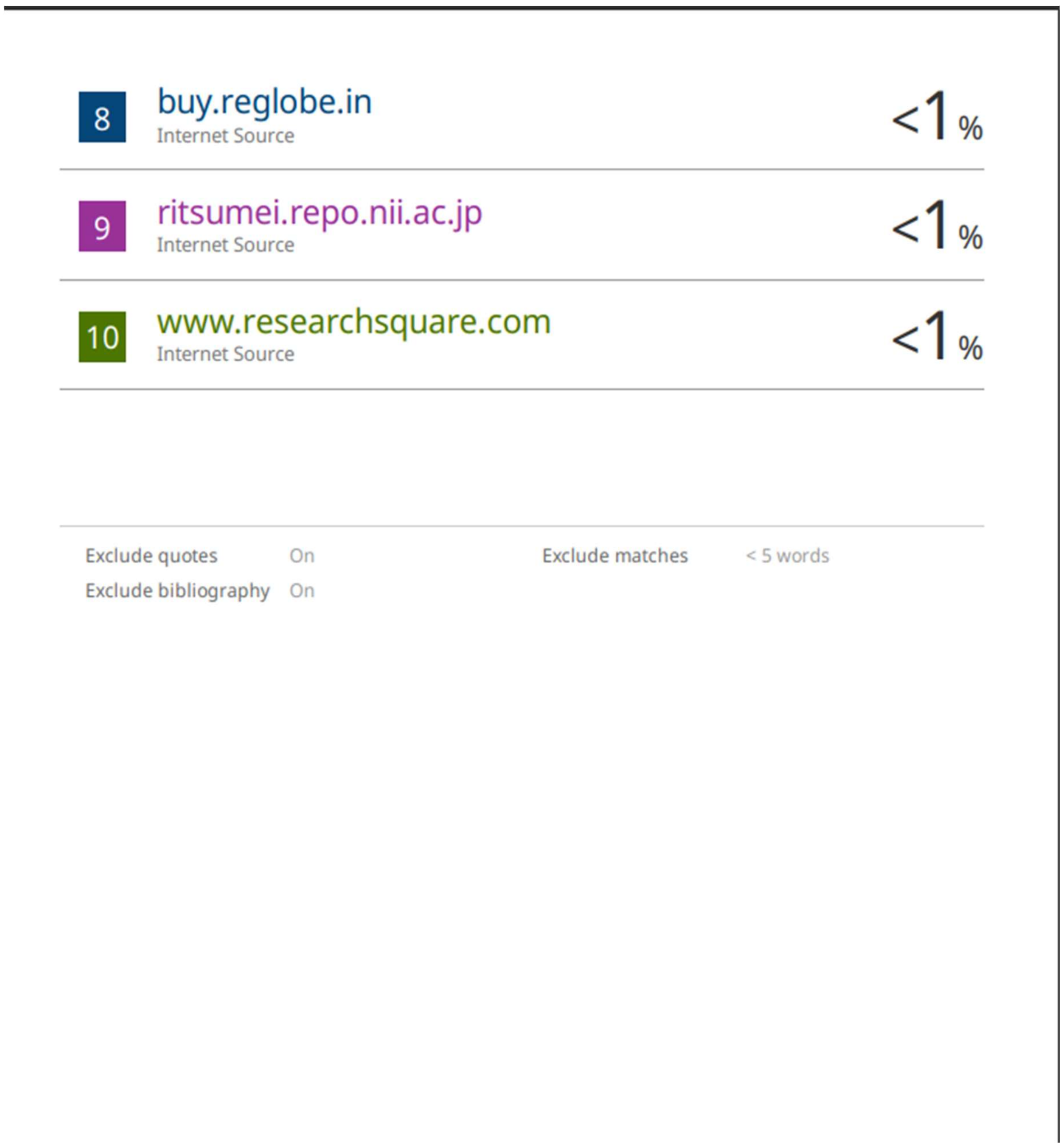
Intelligent System for Identification of Malicious Websites

ORIGINALITY REPORT

7%	3%	1%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Central Lancashire Student Paper	4%
2	apps.dtic.mil Internet Source	1%
3	J Venkatesh, V Vetriselvi, Ranjani Parthasarathi, G Subrahmanya VRK Rao. "Identification and isolation of crypto ransomware using honeypot", 2018 Fourteenth International Conference on Information Processing (ICINPRO), 2018 Publication	1%
4	ijarcce.com Internet Source	1%
5	Submitted to University of Greenwich Student Paper	<1%
6	www.coursehero.com Internet Source	<1%
7	www.grin.com Internet Source	<1%



8	buy.reglobe.in Internet Source	<1 %
9	ritsumei.repo.nii.ac.jp Internet Source	<1 %
10	www.researchsquare.com Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		