



Dissertation on
Detecting Phishing URLs Using ML Techniques

Submitted in partial fulfilment of the requirements for the award of degree of

**Bachelor of Technology
in
Computer Science & Engineering**

UE17CS490B – Capstone Project Phase - 2

Submitted by:

Malla Rishika	PES1201700162
Gagana R	PES1201701618
Niveditha C U	PES1201701640
Keerthi Priya P	PES1201701643

Under the guidance of

S.Nagasundari
Associate Professor
PES University

January - May 2021

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

Detecting Phishing URLs Using ML Techniques

is a bonafide work carried out by

**Malla Rishika
Gagana R
Niveditha C U
Keerthi Priya P**

**PES1201700162
PES1201701618
PES1201701640
PES1201701643**

in partial fulfilment for the completion of eighth semester Capstone Project Phase - 2 (UE17CS490B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2021 – May. 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature
S.Nagasundari
Associate Professor

Signature
Dr. Shylaja S S
Chairperson

Signature
Dr. B K Keshavan
Dean of Faculty

External Viva

Name of the Examiners

Signature with Date

1. _____

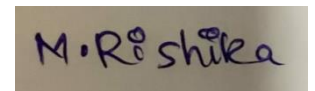
2. _____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled “**Detecting Phishing URLs Using ML Techniques**” has been carried out by us under the guidance of S.Nagasundari, Associate Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology in Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1201700162

Malla Rishika

Handwritten signature of M. Rishika in blue ink on a light background.

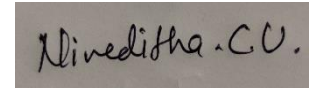
PES1201701618

Gagana R

Handwritten signature of Gagana R in blue ink on a light background.

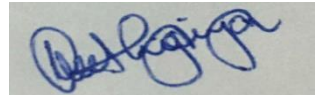
PES1201701640

Niveditha C U

Handwritten signature of Niveditha C U in blue ink on a light background.

PES1201701643

Keerthi Priya P

Handwritten signature of Keerthi Priya P in blue ink on a light background.

ACKNOWLEDGEMENT

We would like to express my gratitude to Associate Professor S.Nagasundari and Professor Prasad.B.Honnavalli , Director of C-ISFCR and C-IOT, Department of Computer Science and Engineering, PES University, for their continuous guidance, assistance, and encouragement throughout the development of this UE17CS490B - Capstone Project Phase – 2.

We are grateful to the project coordinators, Prof.Silviya Nancy and Prof.Sunitha for organizing, managing, and helping with the entire process.

We take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

As the internet is revolutionizing with different technologies every day, it is also increasing threats rapidly. Every user using the internet can be vulnerable to many different types of cyber attacks. Updating and enhancing the security methods with modern solutions is the need of the hour. Phishing is one such common security attack where millions of users are vulnerable to stealing sensitive and confidential information. Phishing attacks imitate the behaviour of a legitimate website which are intended to steal usernames, passwords, health information, contact details and any other kinds of private information. It is challenging and a tedious task to distinguish legitimate websites from phishing websites as attackers innovatively design websites which are very similar to actual websites.

This project aims to provide a method to mitigate the risk of phishing attacks and alert the user beforehand. This project offers an intelligent machine learning approach to detect phishing websites. The system acts as an extension to the browser and notifies the user in case of any phishing activity detected.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	INTRODUCTION	01
2.	PROBLEM STATEMENT	02
3.	LITERATURE REVIEW	03
	3.1 Comparison of Adaboost with MultiBoosting for Phishing Website Detection.	03
	3.1.1 Observation	
	3.1.2 Conclusion and Future Work	
	3.2 WC-PAD: Web Crawling based Phishing Attack Detection.	04
	3.2.1 Observation	
	3.2.2 Conclusion	
	3.3 Intelligent Methods for Accurately Detecting Phishing Websites.	05
	3.3.1 Observation	
	3.3.2 Conclusion	
	3.4 Using Lexical Features for Malicious URL Detection--A Machine Learning Approach	06
	3.4.1 Observation	
	3.4.2 Conclusion	
	3.5 Malicious URL detection using machine learning: A survey	08
	3.5.1 Observation	
	3.5.2 Conclusion	
	3.6 Detecting phishing websites using data mining	10
	3.6.1 Observation	
	3.6.2 Conclusion	
	3.7 Fresh-phish : A Framework for auto-detection of phishing websites	11

3.7.1 Observation	
3.7.2 Conclusion	
3.8 A Feature Selection Comparative Study for Web Phishing Datasets	13
3.8.1 Observation	
3.8.2 Conclusion	
3.9 Intelligent rule-based phishing websites classification	14
3.9.1 Observation	
3.9.2 Conclusion	
3.9.3 Future Works	
4. PROJECT REQUIREMENTS SPECIFICATION	15
5. SYSTEM DESIGN	17
6. PROPOSED METHODOLOGY	19
7. IMPLEMENTATION	21
8. RESULTS AND DISCUSSION	34
9. CONCLUSION AND FUTURE WORK	38

REFERENCES

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

LIST OF FIGURES

Figure No.	Title	Page No.
6.1	Data Flow Diagram	16
6.2	System Design	17
7.1	Module Diagram	19

CHAPTER-1

INTRODUCTION

Phishing is a fake endeavour to acquire sensitive data or information which could be usernames, passwords and bank account information , by disguising oneself as a dependable object in an electronic correspondence. Usually done by texting,email caricaturing, and text informing, phishing frequently fools internet users to enter individual data at a deceiving site which matches the appearances of the legitimate site.

Financial services such as banking are now easily accessible through the Internet to make people's lives simple. It is therefore very important that the safety and security of such facilities be maintained. Phishing is one of the main challenges to web security. By masquerading as a real website or service over the internet, phishing is the technique of stealing user credentials. An ideal solution to the problem of phishing detection is machine learning algorithms that make a device learn new patterns from data. While in recent years there have been several papers that have attempted to detect phishing attacks using machine learning, we plan to go a first step further and develop a software tool that can be easily deployed to detect phishing attacks in end-user systems. Features are extracted and a dataset containing both phishing and legitimate website URLs is created.

CHAPTER 2

PROBLEM STATEMENT

Nowadays, cybercrimes are increasing drastically along with the usage of technology. Cybercrimes have been gradually increasing in fields such as online-transactions, Electronic commerce, social websites. So, it is important to have sophisticated methods to detect and prevent these cybercrimes.

Phishing is one of the most extensively used methods by attackers. Phishing is an attack where attackers send malicious URLs or emails to users and tricks them into providing their sensitive information. As there is a huge sector of online customers who purchase products online and make credit card transactions through various websites, the probability of a phishing attack is very high. There are plenty of websites who ask users to enter confidential information like user-id , password, bank account information ,etc. often for malicious reasons. Such websites are known as phishing websites.

Phishing attacks are exploited by attackers to install ransomware on a targeted machine, by tricking the particular user into downloading an attachment from a malicious URL. Using phishing, attackers can spy on a user and can steal information from their systems.

As cybercrimes are increasing very rapidly because of phishing URLs, it is important to prevent phishing attacks and let users be aware of attacks beforehand. By preventing phishing attacks, a significant amount of cybercrimes can be prevented over the internet.

CHAPTER 3

LITERATURE SURVEY

3.1 Subasi, Abdulhamit, and Emir Kremic. "Comparison of Adaboost with MultiBoosting for Phishing Website Detection." Procedia Computer Science 168 (2020): 272-278.

Observation:

This paper aims to classify websites as legitimate or Phishing and compare Adaboost with Multi-boosting Models to classify phishing website urls. Initially, they have created a dataset considering many different features like IP address, shortening services, Website Traffic, etc.

Then they have implemented two Machine learning Models. One among them is Adaboost. Here they have assigned equal weight to every instance of the training data. And classifier is to be formed on this data and based on the classifier's output, then have assigned weights again. And the process is continued i.e weights of the instances are augmented or reduced and this happens continuously. The next method implemented is Multi Boosting. Multiboosting combines AdaBoost and Wagging. Wagging uses training instances with different weights. Since the training instances have a chance of forming successive training sets, hence wagging assigns random weights to each training set.

They have obtained the following results. Adaboost gained the highest classification accuracy, with Random Forest and with SVM with k-NN, and ANN while Multi boosting achieved a good accuracy with SVM and Random Forest, with k-NN.

Conclusion and Future work:

They have concluded that ensemble way of classifying can enhance the performance of the models in terms of accuracy. Their outcome reveals that by using ensemble models they can detect phishing webpages with an accuracy of 97.61%. Their future work is to remove the dependency on webpage content.

3.2 Nathezhtha, T., D. Sangeetha, and V. Vaidehi. "WC-PAD: Web Crawling based Phishing Attack Detection." In the 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1-6. IEEE, 2019.

Observation:

The aim of this paper is to detect phishing websites using the Web Crawler-based phishing attack detector called (WC-PAD) three-phase attack detection. Techniques used in this paper are DNS Blacklist, Web Crawler, and Heuristic Analysis.

Rule based filters: They are combinations of syntax checkers, URLs, Keywords, IP address etc. By using these rules, they generated rules for detecting the phished emails. These are updated consistently for detection of URLs. It is a three phase phishing attack detection approach. Those three phases are DNS blacklist, Heuristic based approach, Web crawler based approach.

DNS Blacklist: This Blacklist includes the IP address which was involved in spam activities. This list has to be updated frequently. This takes the IP address from the domain and compares it with the Blacklist. An warning is sent to the user if a match is identified. Else it will proceed with the next phase.

Web Crawler: This starts crawling all the pages and all the interconnected links. This goes from one link to another until all the WebPages are crawled. This was proposed to find faults in web indexing. It warns the user if any of the links or web pages don't work.

Heuristic Analysis takes Web content features, URL features, Web traffic features.

Web content features Analysis: This is programmed to crawl through all the contents of a web page. Based on these contents, it classifies web pages to be Legitimate or Phishing.

Web Traffic Analyzer: This considers Google page Rank, total number of visits, Average visit duration in each page, Pages per visit. The Website is classified as phishing or legitimate based on these features.

Conclusion:

Repeatedly used Phishing URLs are detected quickly and easily in DNS Blacklist Technique. Less Frequently used phishing websites are detected in the other phases of WC-PAD.

3.3 Abuzurairq, Almaha, Mouhammd Alkasassbeh, and Mohammad Almseidin. "Intelligent Methods for Accurately Detecting Phishing Websites." In 2020 11th International Conference on Information and Communication Systems (ICICS), pp. 085-090. IEEE, 2020.

Observation:

This paper discusses different methods to distinguish between phishing and legitimate sites.

1. Content Based Approach: This paper talked about an approach which identifies fake web pages by checking the image of logo. At first, the logo is extracted from the downloaded resources of webpage. Next, the identity of image is searched using google search which is name of the domain. Therefore a comparison is done between the domain retrieved and the one from website.

2. Heuristic Based Approach: This paper considers heuristics features extracted from URLs and website rank. The author presented an hybrid machine learning approach. The unwanted data can be best handled by K-nearest neighbours and later SVM(Support Vector Machine) can be used as robust classifier.

The data set used in this paper is composed of equal number of phishing websites and legitimate websites. The html source code and url is utilized for extraction of attributes. In this paper, Infogain and Relief-F techniques are applied for feature selection. The Used Algorithms are Logistic, Random forest, Bagging, J48, Naïve Bayes, and Multilayer perceptron.

Conclusion:

This paper concludes that best accuracy is given by Random forest algorithm.

3.4, Joshi, Apoorva, Levi Lloyd, Paul Westin, and Srini Seethapathy. "Using Lexical Features for Malicious URL Detection--A Machine Learning Approach." arXiv preprint arXiv:1910.06277 (2019).

Observation:

This paper proposes how only lexical features of URLs can be used to classify them as phishing URL or legitimate URL. By using purely lexical features, ML models can give verdict if URL is phishing or not on the internet very quickly.

The procedure followed:

Dataset collection: The dataset consists of ~50 lakh URLs collected from different resources, which includes Openphish, internal FireEye and Alexa whitelists sources, to obtain both Legitimate URLs and phishing URLs.

Feature extraction: Building the classification model starts with extraction of features from the URLs. 23 different lexical features are extracted from URLs that could be useful in differentiating legitimate URLs from malicious URLs. Correlation was used to find interdependencies between features, which is helpful in dimensionality reduction. Along with 23 lexical features, 1000 trigram-based features were considered to produce a dataset with 1023 columns.

Data Modelling: URL strings are very noisy and unstructured. So, it is better to use a model which is not very sensitive to small changes in URLs. This paper does a comparison between different models including Random forest, Gradient Boost, AdaBoost, Logistic Regression, Naïve Bayes. The

comparison showed that the Random forest was a good fit for classification of URLs. Random forest with decision tree estimators was proven to be the best ML model for classification of URLs.

Metrics: Metrics such as Accuracy, FNR , FPR were considered to calculate performance of models and for comparison between the models.

ML model built by considering union of 1000 trigram-based features and lexical features for feature extraction was compared with the model built by considering lexical features. Model with the combination of 1000 trigram-based features and lexical features achieves more accuracy than a model with only lexical features.

Conclusion:

This paper describes Random forest classifier considering only lexical feature- based features can classify Malicious and legitimate URLs. Random forest gives 92% accuracy with 0.38% FNR, which evidently says by using only lexical features, lightweight systems can generate the verdict for URLs in very less amount of time.

3.5 Sahoo, Doyen, Chenghao Liu, and Steven CH Hoi. "Malicious URL detection using machine learning: A survey." arXiv preprint arXiv:1701.07179 (2017).

Observation:

This paper provides a detailed survey and structural understanding of different Machine learning techniques and features extraction methods helpful in detecting Malicious URLs. And it describes various features extraction methods that can be used while building a model.

There are different techniques used to classify Malicious URLs and legitimate ones.

1. **Blacklisting:** Blacklisting is the most commonly used and simple technique for detecting phishing URLs, where a database of already identified malicious URLs are going to be stored. Whenever a new URL is found, that URL is going to be looked up in the database, this URL exists then it is considered as a malicious URL otherwise as a legitimate one.

2. **Heuristic approach:** This is an extension of the blacklisting method, where the logic is to create a “blacklist of signatures”. When a common attack is identified, a signature is going to be attached to this attack type. So when an intrusion detection system scans a webpage, if a suspicious activity is identified then it is going to be brought into notice.

3. **Machine Learning Approach:** This approach tries to analyse the data of a URL and its corresponding webpage and tries to find patterns from URL and web page. By selecting good feature representations of URLs and extracting them, a better ML model can be built with higher accuracy.

The features considered in feature extraction have a large impact on the performance of ML models.

Feature extraction methods:

1. **Lexical features:** These features are acquired from the properties of the URL string. A URL can be classified based on the looks of the URL. Some of them are length of every single component of the URL, count of special characters, count of letters, count of digits, length of the URL.

2. **Host-based features:** These are the features acquired by host-name properties of the URLs. Some features such as IP address properties, Domain name properties, Whois information. Host-based features can be very helpful for classification of URLs.

3. **Content-based features:** Content-based features can be obtained by downloading and extracting the content of webpages. These are “heavyweight” features as a large amount of content is going to be extracted from web pages. These could be HTML based features or JavaScript based features. Some of them are Word count of the document, Length of the document, Distinct word count etc. Javascript features like count of eval() and unescape() functions can be considered, as these functions can be used to encrypt malicious code in a webpage.

4. Other features like Page-based features such as Page rank, Alexa rank and Link Popularity can be very helpful in detecting phishing URLs. Link Popularity is listed by looking at incoming links directed from other websites. It may have a great impact in classification of URLs.

Conclusion:

Detection of phishing URLs is a very important part of cybersecurity applications. A systematic elaboration of phishing URL detection from a machine learning perspective is defined in this paper. This paper describes the requirements and challenges for development of Malicious URL Detection.

3.6 Thaker, Mehek, Mihir Parikh, Preetika Shetty, Vinit Neogi, an Shree Jaswal. "Detecting phishing websites using data mining." In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 1876-1879. IEEE, 2018.

Observation:

This paper mainly discusses phishing website detection using data mining techniques. The techniques discussed in the paper are Black Listing: Whenever a user enters a url, it is checked whether it is present within the black list. The user is alerted if the blacklist contains an url
Similarity based techniques:The design of the normal website is compared with the malicious website to measure similarity metrics which are based on the layout,block-level and overall style. It lacks at covering most of the features of newly launched sites. Content based techniques:These techniques involve checking the webpage content in order to decide if a website is genuine or fake.This methods give enough accuracy and false alarms are low.

Heuristic based approach:This examines features of URL,HTML,Traffic of website,Search Engine.There are some predefined rules for phishing websites and a website is declared as phishing if it matches the rules.

The proposed model is a cloud-based model for detection of malicious sites. Random forest classifier is used as training algorithm for the ML Model

Conclusion:

Random Forest Model is the best classifier with highest accuracy.

3.7 Shirazi, Hossein, Kyle Haefner, and Indrakshi Ray. "Fresh-phish : A Framework for auto-detection of phishing websites." In 2017 IEEE International conference on information reuse and integration(IRI), pp.137-143. IEEE,2017.

Observation:

Objective of this paper is to detect Phishing websites using a python-based open-source framework called Fresh Phish. The Techniques used in this paper are TensorFlow, Gradient Descent, Adagrad, Adadelta, SVM Linear.

Five different features were used by them to build their dataset. These characteristics are divided into five distinct categories,

- URL Based
- DNS Based
- External Statistics
- HTML Based
- JavaScript Based

URL Based: Focused on the Features of URL like whether URL contains IP address, '@', '//, dots in domain part, length, shortening services etc. These features help in identifying the website as Phishing or Legitimate.

DNS Based: Features like how long the registration is valid and when the domain was registered. These features help in detecting Phishing websites.

External Statistics: External Statistics like Alexa rank, Google page rank, if the website is available in Google's record are used for classifying URL.

HTML Based: whether the website contains favicons, iframes, and in cases the pictures and JavaScript has the same root URL as the server's website. Such features assist in determining whether or not a website is malicious.

JavaScript Based: Mouse over methods, disabling right clicks, pop-up-windows with text box are the features of JavaScript that helps in identifying the website as Legitimate or not.

Classifiers: Four classifiers are implemented using scikit-learn and Tfcontrib python libraries. They built DNN (Deep neural network) using TensorFlow and Tfcontrib. For optimization GradientDescent, Adagrad, and Adadelata are used.

Conclusion :

The accuracy was around 90% using Gaussian SVM for fresh phish dataset. They also found TensorFlow took longer to train and was more accurate than SVM.

3.8 Sharma, Suhas R., Rahul Parthasarathy, and Prasad B. Honnavalli. "A Feature Selection Comparative Study for Web Phishing Datasets." In 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1-6. IEEE, 2020

Observation:

This paper mainly focuses on techniques to select best features based on different parameters like F1_Score and Execution time.

The Feature Selection techniques discussed in paper are

1. Chi-Squared Techniques: This technique is based on the relation between two features. If the dependency between two features is very less, the chi-squared value is small because the expected and observed value converge. Thus, if chi-squared value is high, dependency is high which implies the features selected have high chi-squared value.

2. Keiser-Meyer-Olkin (KMO): This feature is a strategy to decrease dimensions based on how fit the “information”. This is mainly the proportion of extent of “variable fluctuation”. If the fluctuation is less, information is more reasonable for investigation of factors.

3. Recursive Feature Elimination (RFE): The technique is to filter out the weakest features so that the model is best suited. This can be accomplished by taking a smaller set of features in loops until the required number of features are acquired.

4. Pearson Correlation: This technique is mainly based on covariance that is the relationship among variables.

Conclusion:

The Random Forest Classifier produces good accuracy when the above discussed techniques are employed.

3.9 Mohammad, Rami M., Fadi Thabtah, and Lee McCluskey. "Intelligent rule-based phishing websites classification." IET Information Security 8, no. 3 (2014): 153-160.

Observation:

The paper presents data mining as a solution in identifying and classifying Phishing websites. In this paper they have developed a software tool which will extract a group of features automatically and these extracted attributes are considered in detecting phishing websites using rules extracted from different rule-induction algorithms. They have derived different features of phishing websites and then categorised them into different classes based on their effect within a webpage. Initially they have extracted features and associated them with a value corresponding to the ratio of that feature in the dataset which indicates them how prominent is that feature in a website. They have considered Abnormal based features, Address bar based features, Javascript and HTML based features.

They have analyzed seventeen distinct features that distinguish phishing webpages from legitimate ones and created a new rule for each feature. And they found that only nine features were more reliable in detecting phishing websites from the legitimate ones.

Conclusion: Features have been extracted without any interference with the users using inbuilt developed tools and explored these rules and developed a new rule for each attribute and have performed frequency analysis for each attribute.

Future Work: They want to further use the rules developed by the distinct algorithms to develop a software that is incorporated with a web browser to detect phishing websites in real time and alert the internet user about the webpage.

CHAPTER 4

PROJECT REQUIREMENTS SPECIFICATION

Functional Requirements

This may include,

1. The webpage content of the URL entered should be scrapped.
2. The URL should be decomposed into attributes and should be fed into the ML model.
3. The system should process the output from the backend and alert the user if it's a phishing URL.

Software Requirements

- Python with required libraries
- Anaconda navigator
- Chrome
- Javascript
- PHP
- HTML
- CSS

Hardware Requirements

- Working PC
- RAM size \geq 8GB

Data Requirements

Source : The dataset considered in this project is taken from Kaggle, which consists of URLs and corresponding target values.

Size : The dataset consists of 10,000 URLs, in which 5,000 URLs are phishing and 5,000 URLs are legitimate.

Features : The following features are extracted from URLs for dataset formation. Url having ip address, Length of URL, TinyURL, At the rate count, Redirection using '//', Hyphen in domain name, Sub domain, Age of domain, Web traffic, Iframe, Mouseover, Right click, Forwarding, Http domain, Domain registration, Ssl final state, Request url.

Non-Functional Requirements

Performance Requirement

Efficiency:

The URL entered should be classified in near real time. The model should not produce a high False Positive Rate and high False Negative Rate.

Privacy:

Privacy can be a concern since certain headers are evaluated during feature extraction.

Availability:

We will add our software as a chrome extension, so the software is going to be used only in chrome browser.

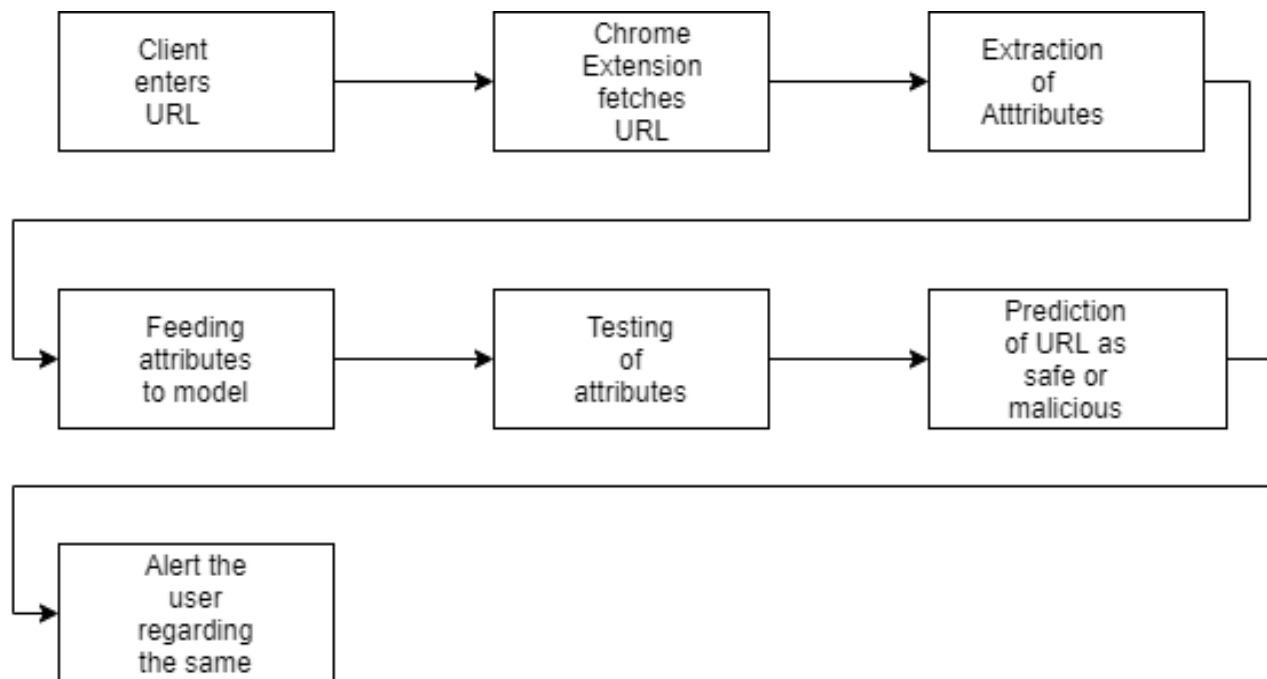
Portability:

Software is not portable on mobile devices.

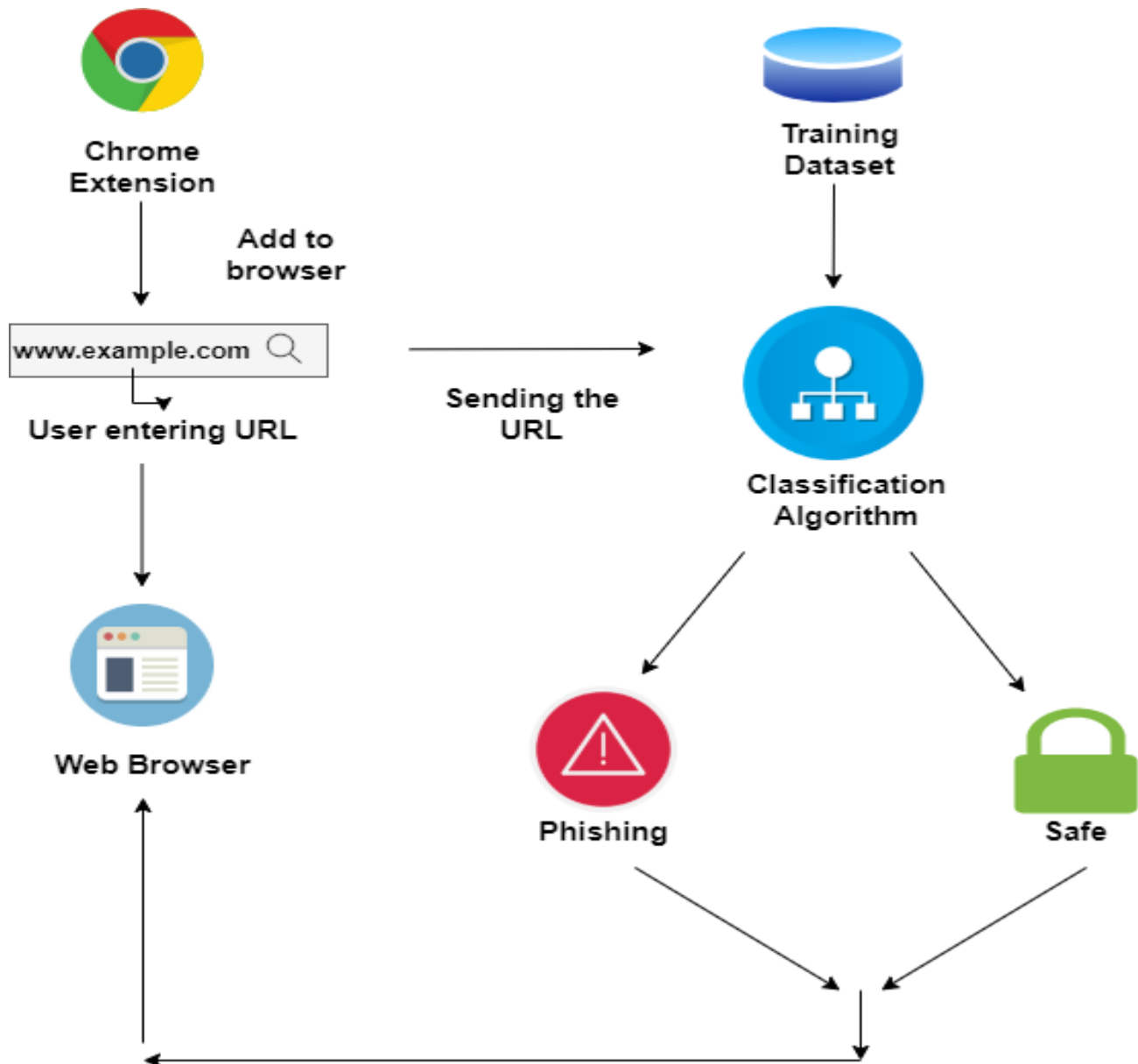
CHAPTER 5

SYSTEM DESIGN

In the system, on the client side, a user enters the URL. The chrome extension which is added to the browser fetches the URL and numerous attributes of the webpage are extracted. These attributes are fed as an input which is a test data for the classifier. The classifier will predict the output as either safe or unsafe. The user will be alerted if the webpage is malicious one.



6.1 Data Flow Diagram



6.2 System Design

CHAPTER 6

PROPOSED METHODOLOGY

The system which is a chrome extension is divided into backend and frontend (plugin). The frontend plugin consists of a JavaScript file which extracts features and sends them to a PHP file. The backend includes dataset which is pre-processed, a python file which has Random forest as a training module and PHP file which links JavaScript and python file at the backend. The plugin also consists of HTML, CSS and manifest files for the user interface as a part of chrome extension.

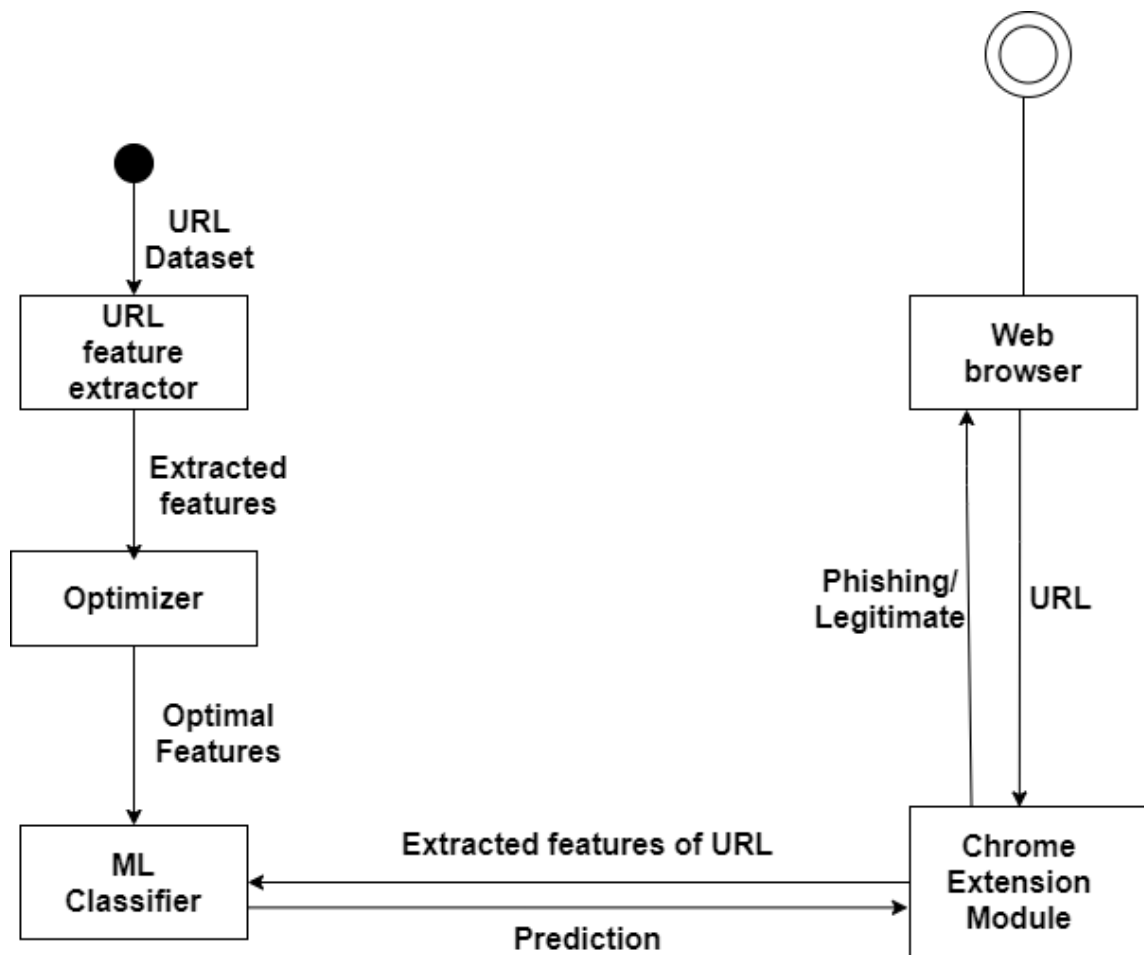
The input and output to each module of the system is described in this section.

Feature Extraction Phase: This module takes a URL dataset with target value as input and outputs a dataset with the features. The features that we have extracted are URL Length, Presence of IP Address, Presence of Shortening Service, having @ Symbol , ‘//’ Redirection, Prefix-Suffix , having Subdomain, SSL final State, domain Registration Length , favicon, https token, Request URL, URL Anchor, Links in Tags, Server Form Handler, Submitting to Email, Abnormal URL, Redirect, on-mouseover, Disabling Right click, Pop-Up Window, Iframe Redirection, Age of Domain, DNS Record, Web Traffic, Links pointing to Page. After extracting features, Particle Swarm optimization is performed where it selects optimal features. After performing PSO, out of 26 features, 22 features were considered optimal features. And then dataset with 22 features have been extracted and created a dataset.

Random Forest Training Phase: In this Phase, the dataset with all the extracted features is taken as an input to the Random Forest Classifier and the classifier trains the model with the dataset and then is tested and has obtained an accuracy of 97%. And this model is converted to a joblib format so as to reduce the prediction time.

Chrome Extension Phase: This module includes JavaScript file along with APIs developed by chrome which fetches the URL from the address bar, performs feature extraction and sends the feature extraction vector to the PHP file. After receiving the feature extraction vector, the PHP file inputs to the python file consisting of ML classifier which is already exported in a pickle format and

written to a file in disk. This aids in reducing the burden of executing the Random forest Classifier for every prediction. After predicting, the output is forwarded to the chrome extension.



7.1 Module Diagram

CHAPTER 7

IMPLEMENTATION

Feature Extraction

These below features considered for the Feature Extraction

1. Age of Domain

```
def age_of_domain(url):  
    try:  
        w = whois(url)  
        start_date = w.creation_date  
        current_date = datetime.datetime.now()  
        age = (current_date - start_date[0]).days  
        if (age >= 180):  
            return -1  
        else:  
            return 1  
    except Exception as e:  
        return 1
```

2. HTTPS token in URL

```
def http_domain(url):  
    try:  
        r = requests.get('http://' + url, timeout=10)  
        if 'https' in r.url:  
            return -1  
        else:  
            return 1  
    except:  
        return 1
```

3. IP address in URL

```
def url_ip(url):  
    try:  
        a=re.match('((http|https): (//))?.*\d+\.\d+\.\d+\.\d+\.*',url)  
        if a is None:  
            return 0  
        else:  
            return 1  
    except:  
        return -1
```

4. Length Of the URL

```
def getLength(url):  
    length=len(url)  
    if(length<54):  
        return -1  
    elif(54<=length<=75):  
        return 0  
    else:  
        return 1
```

5. Redirection URL

```
def redirection(url):  
    if 'http' not in url and 'https' not in url:  
        url='http://' +url  
    pos = url.rfind('///')  
    if pos > 6:  
        if pos > 7:  
            return 1  
        else:  
            return -1  
    else:  
        return -1
```

6. '@' symbol in URL

```
def at_the_rate_count(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
        return -1
    else:
        return 1
```

7. Prefix-Suffix separated '-' in URL domain

```
def prefixSuffix(url):
    if 'http' not in url and 'https' not in url:
        url='http://' +url
    if '-' in urlparse(url).netloc:
        return 1
    else:
        return -1
```

8. Shortening of URL

```
def tinyURL(url):
    shortening_services = r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs| " \
        r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com| " \
        r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us| " \
        r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|db\.tt| " \
        r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|q\.gs|is\.gd| " \
        r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzzurl\.com|cutt\.us|u\.bb|yourls\.org|x\.co| " \
        r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|lurl\.com|tweez\.me|v\.gd| " \
        r"tr\.im|link\.zip\.net"
    match = re.search(shortening_services, url)
    if match:
        return 1
    else:
        return -1
```

9. Presence of dots in subdomain

```
def sub_domain(url):  
    subDomain, domain, suffix = extract(url)  
    if (subDomain.count('.') == 1):  
        return 0  
    elif (subDomain.count(".") == 2):  
        return 1  
    else:  
        return -1
```

10. Domain Registration Length:

```
def domain_registration(url):  
    try:  
        w = whois.whois(url)  
        updated = w.updated_date  
        exp = w.expiration_date  
        length = (exp[0]-updated[0]).days  
        if(length<=365):  
            return 1  
        else:  
            return -1  
    except:  
        return -1
```

11. SSL Final State

```
def SSLfinal_State(url):
    try:
        #check wheather contains https
        if (regeX.search("https",url)):
            usehttps = 1
        else:
            usehttps = 0

        #getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        content = ssl.create_default_context()
        sct = content.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_auth = str(issuer['commonName'])
        certificate_auth = certificate_auth.split()
        if (certificate_auth[0] == "Network" or certificate_auth == "Deutsche"):
            certificate_auth = certificate_auth[0] + " " + certificate_auth[1]
        else:
            certificate_auth = certificate_auth[0]

        trusted_auth = ['Comodo', 'Symantec', 'GoDaddy', 'GlobalSign', 'DigiCert', 'StartCom', 'Entrust', 'Verizon', 'Trustwave', 'Uninet', 'Bypass', 'QuoVadis', 'Deutsche Telekom', 'Network Solutions', 'SwissSign', 'IdenTrust', 'Secom', 'TWCA', 'GeoTrust', 'Thawte', 'Doster']

        #getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[2])
        endingYear = int(endingDate.split()[2])
        age_of_certificate = endingYear-startingYear

        #checking final conditions
        if ((usehttps==1) and (certificate_auth in trusted_auth) and (age_of_certificate>=1) ):
            return -1 #legitimate
        elif ((usehttps==1) and (certificate_auth not in trusted_auth)):
            return 0 #suspicious
        else:
            return 1 #phishing

    except Exception as e:
        return -1
```


12. Multimedia linked to same domain of Webpage

```
def request_url(url):
    if 'http' not in url and 'https' not in url:
        url='http://'+url
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)
        linked_to_same = 0
        avg = 0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if (websiteDomain == imageDomain or imageDomain == ''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)
        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if (websiteDomain == vidDomain or vidDomain == ''):
                linked_to_same = linked_to_same + 1
        linked_outside = total - linked_to_same
        if (total != 0):
            avg = linked_outside / total
        if (avg<0.22):
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0
```

13. Links in tags

```
def Links_in_tags(url):
    if 'http' not in url and 'https' not in url:
        url='http://'+url
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        no_of_meta = 0
        no_of_link = 0
        no_of_script = 0
        anchors = 0
        avg = 0
        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta + 1
        for link in soup.find_all('link'):
            no_of_link = no_of_link + 1
        for script in soup.find_all('script'):
            no_of_script = no_of_script + 1
        for anchor in soup.find_all('a'):
            anchors = anchors + 1
        total = no_of_meta + no_of_link + no_of_script + anchors
        tags = no_of_meta + no_of_link + no_of_script
        if (total != 0):
            avg = tags / total
        if (avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0
```

14. Presence of mailto function

```
def email_submit(url):  
    if 'http' not in url and 'https' not in url:  
        url='http://'+url  
    try:  
        opener = urllib.request.urlopen(url).read()  
        soup = BeautifulSoup(opener, 'lxml')  
        if(soup.find('mailto:')):  
            return 1  
        else:  
            return -1  
    except:  
        return 0
```

15. Presence of Hostname

```
def abnormal_url(url):  
    if 'http' not in url and 'https' not in url:  
        url='http://'+url  
    domain = urlparse(url).netloc  
    if url.find(domain)==-1:  
        return 1  
    else:  
        return -1
```

16. Presence of mouseover function in script

```
def mouseOver(url):  
    if 'http' not in url and 'https' not in url:  
        url='http://'+url  
    try:  
        response = requests.get(url)  
    except:  
        response = ""  
    if response == "" :  
        return 1  
    else:  
        if re.findall("<script>.+onmouseover.+</script>", response.text):  
            return 1  
        else:  
            return -1
```

17. Existence of IFrame

```
def iframe(url):
    if 'http' not in url and 'https' not in url:
        url='http://'+url
    try:
        response = requests.get(url)
    except:
        response = ""
    if response == "":
        return 1
    else:
        if '<iframe>' in response.text or '<frameBorder>' in response.text:
            return 1
        else:
            return -1
```

18. Disabling Right Click

```
def rightClick(url):
    if 'http' not in url and 'https' not in url:
        url='http://'+url
    try:
        response = requests.get(url)
    except:
        response = ""
    if response == "":
        return 1
    else:
        if re.findall(r"event.button ?== ?2", response.text):
            return 1
        else:
            return -1
```

19. Forwarding of Webpage

```
def forwarding(url):
    if 'http' not in url and 'https' not in url:
        url='http://' +url
    try:
        response = requests.get(url)
    except:
        response = ""
    if response == "":
        return 1
    else:
        if len(response.history) <=1:
            return -1
        elif (len(response.history)>=2 and len(response.history)<4):
            return 0
        else:
            return 1
```

20. Appearance of pop up window

```
def popUpwindow(url):
    if 'http' not in url and 'https' not in url:
        url='http://' +url
    try:
        response = requests.get(url)
    except:
        response = ""
    try:
        if re.findall(r"alert\(", response.text):
            return 1
        else:
            return -1
    except:
        return -1
```

21. Count of links pointing to a page

```
def numberoflinks(url):
    if 'http' not in url and 'https' not in url:
        url='http://' + url
    try:
        response = requests.get(url)
    except:
        response = ""
    try:
        number_of_links = len(re.findall(r"<a href=", response.text))
        if number_of_links == 0:
            return 1
        elif number_of_links > 0 and number_of_links <= 2:
            return 0
        else:
            return -1
    except:
        return 0
```

22. Anchors in Source Code

```
def url_of_anchor(url):
    if 'http' not in url and 'https' not in url:
        url='http://' + url
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        anchors = soup.findAll('a', href=True)
        total = len(anchors)
        linked_to_same = 0
        avg = 0
        for anchor in anchors:
            subDomain, domain, suffix = extract(anchor['href'])
            anchorDomain = domain
            if (websiteDomain == anchorDomain or anchorDomain == ''):
                linked_to_same = linked_to_same + 1
        linked_outside = total - linked_to_same
        if (total != 0):
            avg = linked_outside / total
        if (avg < 0.31):
            return -1
        elif avg >= 0.31 and avg <= 0.67:
            return 0
        else:
            return 1
    except:
        return 0
```

23.DNS Record

```
def dns_record(url):
    try:
        answers=dns.resolver.query(url)
        for rdata in answers:
            return 0
    except:
        return 1
```

24. Favicon

```
def get_favicon_url(domain):
    try:
        page = requests.get(domain, timeout=10)
        soup = BeautifulSoup(page.text, features="lxml")
        icon_link = soup.find("link", rel="shortcut icon")
        if icon_link is None:
            icon_link = soup.find("link", rel="icon")
        if icon_link is None:
            return domain + '/favicon.ico'
        return icon_link["href"]
    except:
        return 'None'
```

25. Server Form Handler

```
def SFH(url):
    if 'http' not in url and 'https' not in url:
        url = 'http://' + url
    try:
        response = requests.get(url, timeout=10)
        if response == "":
            return 1
        else:
            soup = BeautifulSoup(response.text, 'html.parser')
            subDomain, domain, suffix = extract(url)
            for form in soup.find_all('form', action=True):
                if form['action'] == "" or form['action'] == "about:blank" :
                    return 1
                elif url not in form['action'] and domain not in form['action']:
                    return 1
                else:
                    return -1
            return -1
    except:
        return 1
```

26. Web Traffic

```
def web_traffic(url):  
    try:  
        url = urllib.parse.quote(url)  
        rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=surl=" + url).read(), "xml").find("REACH")['RANK']  
  
    except TypeError:  
        return 0  
    rank=int(rank)  
  
    if rank < 100000:  
        return -1  
    elif rank == 100000:  
        return 0  
    else:  
        return 1
```

Random Forest with PSO Optimization

```
clf = linear_model.LogisticRegression()  
  
def f_per_particle(m, alpha):  
    total_features = X.shape[1]  
    if np.count_nonzero(m) == 0:  
        X_subset = X  
    else:  
        X_subset = X[:,m==1]  
  
    clf.fit(X_subset, y)  
    P = (clf.predict(X_subset) == y).mean()  
    j = (alpha * (1.0 - P)  
        + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))  
    return j  
  
def f(x, alpha=0.9):  
    n_particles = x.shape[0]  
    j = [f_per_particle(x[i], alpha) for i in range(n_particles)]  
    return np.array(j)
```



```
from datetime import datetime as dt
import time

start = dt.now()
print("Started at: ", str(start))
particleScore = list()
particleSize = list()

options = {'c1': 2, 'c2': 2, 'w':0.3, 'k': 20, 'p':2}

dimensions = X.shape[1]
optimizer = ps.discrete.BinaryPSO(n_particles= 30, dimensions=dimensions, options=options)

cost, pos = optimizer.optimize(f, iters=25, verbose=2)

end = dt.now()
print("Finished at: ", str(end))
total = end-start
print("Total time spent: ", total)

X_selected_features = X[:,pos==1]
print(len(X_selected_features[0]))

X_train, X_test, y_train, y_test = train_test_split(X_selected_features, y, test_size=0.3)

clf2 = RandomForestClassifier(n_estimators =50,max_depth =15)

from sklearn import metrics

clf2.fit(X_train, y_train)
predictions = clf2.predict(X_test)
accuracy = metrics.accuracy_score(y_test, predictions)
print(accuracy)
```

CHAPTER-8

RESULTS AND DISCUSSION

8.1 Dataset For Testing

phish Tank lists out various recently found links of phishing websites. The chrome extension is tested with websites that are listed in phishTank. It should be noted that the plugin is able to detect new phishing sites too. The results are summarised below.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	
id	having_If_URL_Ler	Shortnin	having_7	double_s	Prefix_Si	having_S	SSLfinal	Domain	Favicon	port	HTTPS_	Request_	URL_of_	Links_in	SFH	Submittir	Abnorm:	Redirect	on_mou:	RightClic	popUp/vi	Iframe	age_of_	DNSRec	web_traff	Page_Ri	Google_	Links_pc	Statistic	Result		
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	
2	1	1	1	1	1	1	-1	0	1	1	1	-1	1	0	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	1	1	-1	
3	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	1	-1	1	1	0	-1	
4	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	
5	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	0	-1	1	1	1	1	
6	-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	
7	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	
8	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	-1	-1	1	1	0	1	1	1	1	-1	0	-1	1	1	0	1	-1	
9	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	1	1	1	1	
10	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	
11	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	
12	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	0	-1	-1	
13	-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	1	0	1	-1	
14	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	
15	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	1	
16	1	-1	-1	-1	1	-1	0	0	1	1	1	1	-1	-1	0	-1	1	1	0	1	1	1	1	1	-1	-1	-1	1	0	1	-1	
17	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	-1	
18	1	-1	1	1	1	1	-1	-1	0	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	1	-1	1	1	-1	1	1	-1	-1	
19	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	-1	1	
20	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	0	-1	-1	1	0	-1	
21	1	0	-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	
22	1	0	1	1	1	1	-1	0	1	1	1	1	-1	-1	0	-1	-1	-1	0	1	1	1	1	-1	1	-1	-1	1	0	-1	1	
23	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	-1	1	1	
24	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	
25	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	
26	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	
27	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	
28	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	0	-1	1	0	-1	
29	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	1	-1	1	1	
30	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	0	1	1
31	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	0	1	1
32	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	0	-1	1	
33	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	
34	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	1	-1	1	1	0	-1	1
35	1	0	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	1	1	-1	-1	

8.2 Output in various Stages

This section shows the results obtained during module testing.

8.2.1 Preprocessing

The output of preprocessing module is shown in figure 8.1.

```

===== RESTART: C:\Users\Lenovo\Desktop\CAPSTONE\PHASE-2\check.py =====
['having_IP_Address', 'URL_Length', 'Shortning_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State', 'Domain_registration_length', 'Favicon', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord', 'web_traffic', 'Links_pointing_to_page', 'Result']

Total number of features without PSO 27
=====

```

8.2.2 Training

The output of the training module is shown in figure 8.2

```

===== RESTART: C:\xampp\htdocs\C_E_NEW_PSO\RandomForest.py =====
Accuracy: 0.9701537533916189
F1 Score: 0.9735788630904724
False Postive Rate: 0.033287101248266296
False Negitve Rate: 0.0272
True Negative Rate: 0.9667128987517337
True Positive Rate: 0.9728
=====

```

8.2.3 Classification

The output of the classification is shown in the Plugin UI.



Phishing Detector

The Link:

<https://www.google.com/>

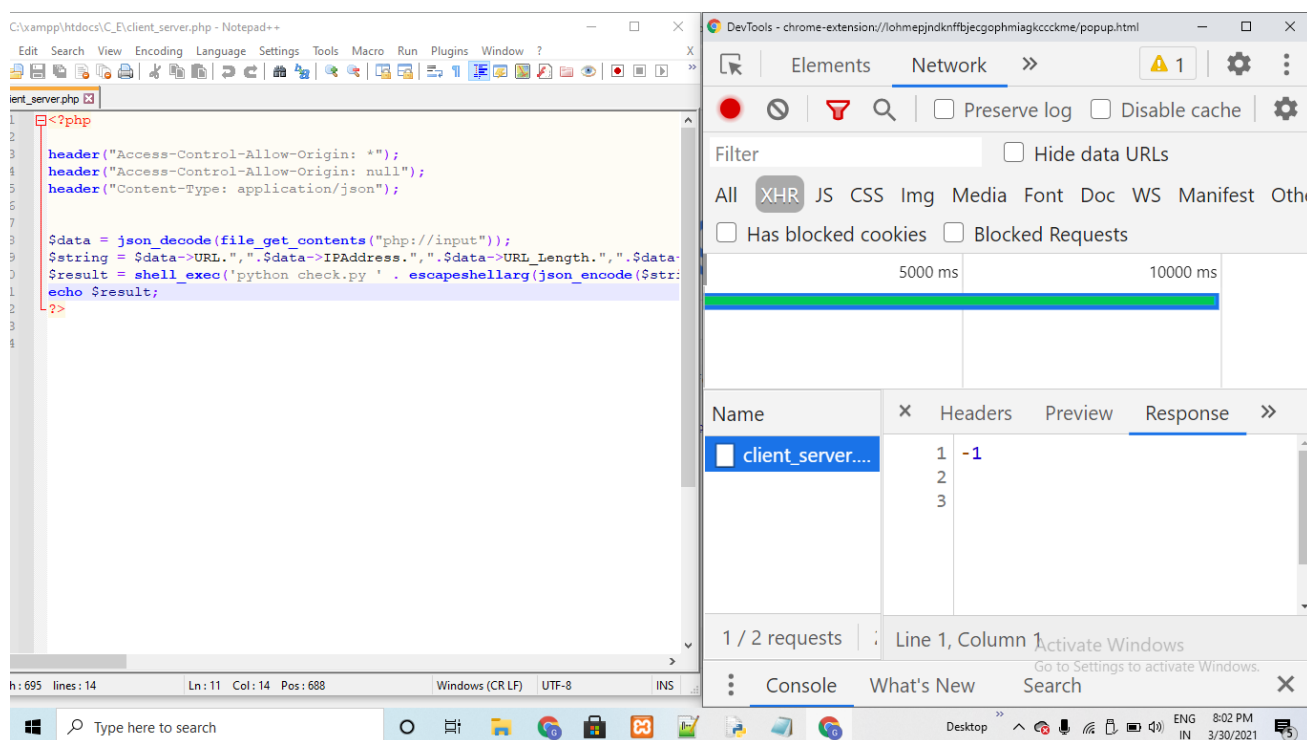
Check if the above link is phishing....



IS IT PHISHING?

The site is legitimate

[Click on me to know more about Phishing!](#)



The screenshot displays a development environment with two windows. The left window is Notepad++, showing a PHP script named `client_server.php`. The script includes headers for Access-Control-Allow-Origin and Content-Type, and a function that decodes a JSON input, extracts URL and IP address information, and executes a shell command to check for a shell escape.

```

1 <?php
2
3 header("Access-Control-Allow-Origin: *");
4 header("Access-Control-Allow-Origin: null");
5 header("Content-Type: application/json");
6
7
8 $data = json_decode(file_get_contents("php://input"));
9 $string = $data->URL.", ".$data->IPAddress.", ".$data->URL_Length.", ".$data->URL_
10 $result = shell_exec('python check.py ' . escapeshellarg(json_encode($string)) . ' ');
11 echo $result;
12
13 ?>

```

The right window is Chrome DevTools, showing the Network tab. A request to `chrome-extension://lohmepjndknftbjecgophmiagkccckme/popup.html` is selected. The request is an XHR (XMLHttpRequest) with a status of 200. The response is visible in the Response pane, showing a JSON array with one element: `["client_server...."]`.

CHAPTER 9

CONCLUSION AND FUTURE WORK

This is an intelligent phishing website detection system which warns the users before getting phished and prevents the users from getting their credentials misused. Machine Learning technique is used in order to achieve high accuracy and rapid detection.

Future Work

Currently, the classifier is trained on 26 features which can be increased further, provided they neither result in loss of privacy nor increase the detection time. The chrome extension can be enhanced to cache the results of frequently visited sites thereby reducing the computation time. Threads can be implemented for feature extraction to execute the features parallel which may result in significant reduction in time. Thus there's a lot of scope for enhancements and improvements making it a more usable solution for phishing detection.

REFERENCES

- [1]Subasi, Abdulhamit, and Emir Kremic. "Comparison of Adaboost with MultiBoosting for Phishing Website Detection." *Procedia Computer Science* 168 (2020): 272-278
- [2]Nathezhtha, T., D. Sangeetha, and V. Vaidehi. "WC-PAD: Web Crawling based Phishing Attack Detection." In the 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1-6. IEEE, 2019.
- [3]Abuzurairq, Almaha, Mouhammd Alkasassbeh, and Mohammad Almseidin. "Intelligent Methods for Accurately Detecting Phishing Websites." In 2020 11th International Conference on Information and Communication Systems (ICICS), pp. 085-090. IEEE, 2020.
- [4]Joshi, Apoorva, Levi Lloyd, Paul Westin, and Srini Seethapathy. "Using Lexical Features for Malicious URL Detection--A Machine Learning Approach." *arXiv preprint arXiv:1910.06277* (2019).
- [5]Sahoo, Doyen, Chenghao Liu, and Steven CH Hoi. "Malicious URL detection using machine learning: A survey." *arXiv preprint arXiv:1701.07179* (2017).
- [6]Thaker, Mehek, Mihir Parikh, Preetika Shetty, Vinit Neogi, an Shree Jaswal. "Detecting phishing websites using data mining." In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 1876-1879. IEEE, 2018.
- [7]Shirazi, Hossein, Kyle Haefner, and Indrakshi Ray. "Fresh-phish : A Framework for auto-detection of phishing websites." In 2017 IEEE International conference on information reuse and integration(IRI),pp.137-143. IEEE,2017.

[8]Sharma, Suhas R., Rahul Parthasarathy, and Prasad B. Honnavalli. "A Feature Selection Comparative Study for Web Phishing Datasets." In 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1-6. IEEE, 2020

[9]Mohammad, Rami M., Fadi Thabtah, and Lee McCluskey. "Intelligent rule-based phishing websites classification." IET Information Security 8, no. 3(2014): 153-160.

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

ML-Machine Learning

FPR- False Positive Rate

FNR-False Negative Rate

URL-Uniform Resource Locator

HTML-Hyper Text Markup Language

CSS-Cascading Style Sheets

PHP-Hypertext Preprocessor