WEB PHISHING DETECTION

Team Id : PNT2022TMID38146

VENKATASUBRAMANIAN S
SIVANRAJA K
PRANAV PRADEEP
SETHUPATHY K
HARIHARAN N

1. ## INTRODUCTION

   a. ## PROJECT OVERVIEW

There are number of users who purchase products online and make payment through various websites. There are multiple websites who ask user to provide sensitive data such as username, password or credit card details etc. often for malicious reasons. This type of websites is known as phishing website. In order to detect and predict phishing website, we proposed an intelligent, flexible and effective system that is based on using classification Data mining algorithm. We implemented classification algorithm and techniques to extract the phishing data sets criteria to classify their legitimacy.

The phishing website can be detected based on some important characteristics like URL and Domain Identity, and security and encryption criteria in the final phishing

detection rate. Once user makes transaction through online when he makes payment through the website our system will use data mining algorithm to detect whether the website is phishing website or not. This application can be used by many E-commerce enterprises in order to make the whole transaction process secure.

Data mining algorithm used in this system provides better performance as compared to other traditional classifications algorithms. With the help of this system user can also purchase products online without any hesitation. Admin can add phishing website url or fake website url into system where system could access and scan the phishing website and by using algorithm, it will add new suspicious keywords to database. System uses machine learning technique to add new keywords into database.

b. PURPOSE

There are numerous algorithms and a wide variety of data types for phishing detection in the academic literature and commercial products. A phishing URL and the corresponding page have several features which can be differentiated from a malicious URL. For example; an attacker can register long and confusing domain to hide the actual domain name **(**Cybersquatting , Typosquatting). In some cases attackers can use direct IP addresses instead of using the domain name. This type of event is out of our scope, but it can be used for the same purpose. Attackers can also use short domain names which are irrelevant to legitimate brand names and don’t have any FreeUrl addition. But these type of web sites are also out of our scope, because they are more relevant to fraudulent domains instead of phishing domains.

URL BASED TECHNIQUE

URL is the first thing to analyse a website to decide whether it is a phishing or not. As we mentioned before, URLs of phishing domains have some distinctive points. Features which are related to these points are obtained when the URL is processed. Some of URL-Based Features are given below.

a. Digit count in the URL

b. Total length of URL

c. Checking whether the URL is Typosquatted or not. (google.com → goggle.com)

d. Checking whether it includes a legitimate brand name or not (apple-icloud-login.com)

e. Number of subdomains in URL

f. Is Top Level Domain (TLD) one of the commonly used one?

The purpose of Phishing Domain Detection is detecting phishing domain names. Therefore, passive queries related to the domain name, which we want to classify as phishing or not, provide useful information to us.

Page-Based Features are using information about pages which are calculated reputation ranking services. Some of these features give information about how much reliable a web site is.

Obtaining these types of features requires active scan to target domain. Page contents are processed for us to detect whether target domain is used for phishing or

not.

2. LITERATURE SURVEY

Phishing Detection: A Literature Survey

a. April 2013
b. IEEE Communications Surveys & Tutorials
c. Project: Mitigation of Phishing Attacks

Authors:

d. Mahmoud Khonji
e. Youssef Iraqi
f. Andy Jones

Abstract and Figures

This article surveys the literature on the detection of phishing attacks. Phishing
attacks target vulnerabilities that exist in systems due to the human factor. Many cyber attacks are spread via mechanisms that exploit weaknesses found in endusers, which makes users the weakest element in the security chain.
The phishing problem is broad and no single silver-bullet solution exists to mitigate all the
vulnerabilities effectively, thus multiple techniques are often implemented to

mitigate specific attacks.
This paper aims at surveying many of the recently proposed phishing mitigation techniques. A high-level overview of various categories of phishing mitigation techniques is also

presented, such as: detection, offensive defense, correction, and prevention, which we belief is critical to present where the phishing detection techniques fit in the overall mitigation process.
Rao et al proposed a novel classification approach that use heuristic-based feature extraction approach. In this, they have classified extracted features into three categories such as URL Obfuscation features, Third-Party-based features, Hyperlink-based features. Moreover, proposed technique gives 99.55% accuracy. Drawback of this is that as this model uses third party features, classification of website dependent on speed of third-party services. Also this model is purely depends on the quality and quantity of the training set and Broken links feature extraction has a Volume 3.

Chunlin et al proposed approach that primarily focus on character frequency features. In this they have combined statistical analysis of URL with machine learning technique to get result that is more accurate for classification of malicious URLs. Also they have compared six
machine-learning algorithms to verify the effectiveness of proposed algorithm which gives 99.7% precision with false positive rate less than 0.4%.

Sudhanshu et al used association data mining approach. They have proposed rule based classification technique for phishing website detection. They have concluded that association classification algorithm is better than any other algorithms because of their simple rule
transformation. They achieved 92.67% accuracy by extracting 16 features but this is not up to mark so proposed algorithm can be enhanced for efficient detection rate.

M. Amaad et al presented a hybrid model for classification of phishing website. In this paper, proposed model carried out in two phase. In phase 1,they individually perform classification techniques, and select the best three models based on high accuracy and other performance criteria. While in phase 2, they further combined each individual model with best three model and makes hybrid model that gives better accuracy than individual model. They achieved 97.75% accuracy on testing dataset. There is limitation of this model that it requires more time to build hybrid mode.

a. EXISTING SYSTEM

The existing system uses the Classifiers, Fusion Algorithm, and Bayesian Model to detect the phishing sites. The classifiers can classify the text content and image content. Text classifier is to classify the text content and Image classifier is to classify the image content. Bayesian model estimates the threshold value. Fusion Algorithm combines the both classifier results and decides whether the site is phishing or not. The performance
of different classifiers based on correct classification ratio, F-score, Matthews's correlation coefficient, False negative ratio, and False alarm ratio. The threshold value will be decided by the developer only. This leads to the problems like false positive and false negative. False positive means, the probability of being a phishing webpage is greater than the threshold value but that webpage is not a phishing webpage. False negative means, the probability of being a phishing webpage is less than the threshold value but that webpage is a phishing webpage. This results the reduction in security levels. The existing system handles the only one kind of phishing attacks. If that was a phishing site then the existing system only warns

the user. The active and passive warnings alone were not sufficient to control the phishing sites. The active warning gives the user options to close the window or displaying the website. The passive warning displays the popup dialog box.

b. REFERENCES

[1] Routhu Srinivasa Rao1 , Alwyn Roshan Pais :Detection of phishing websites using an efficient feature-based machine learning framework :In Springer 2018. Volume 3, Issue 7, September-october-2018 | http:// ijsrcseit.com Purvi Pujara et al. Int J S Res CSE & IT. 2018 SeptemberOctober-2018; 3(7) : 395-399 399.

[2] Chunlin Liu, Bo Lang : Finding effective classifier for malicious URL detection :
InACM,2018.

[3] Sudhanshu Gautam, Kritika Rani and Bansidhar Joshi : Detecting Phishing Websites Using Rule-Based Classification Algorithm: A Comparison : In Springer,2018.

[4] M. Amaad Ul Haq Tahir, Sohail Asghar, Ayesha Zafar, Saira Gillani : A Hybrid Model to Detect Phishing-Sites using Supervised Learning Algorithms :In International Conference on Computational Science and Computational Intelligence IEEE ,2016.

[5] Hossein Shirazi, Kyle Haefner, Indrakshi Ray: Fresh-Phish: A Framework for Auto- Detection of Phishing Websites: In (International Conference on Information Reuse and Integration (IRI)) IEEE,2017.

[6] Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach :In Springer Science+Business Media, LLC, part of Springer Nature 2017.

[7] Bhagyashree E. Sananse, Tanuja K. Sarode : Phishing URL Detection: A Machine Learning and Web Mining-based Approach : In International Journal of ComputerApplications,2015.

[8] Mustafa AYDIN, Nazife BAYKAL : Feature Extraction and Classification

Phishing Websites Based on URL : IEEE,2015.
[9] Priyanka Singh, Yogendra P.S. Maravi, Sanjeev Sharma : Phishing Websites Detection through Supervised Learning Networks : In IEEE,2015.
[10] Pradeepthi. K V and Kannan. A: Performance Study of Classification Techniques for Phishing URL Detection: In 2014 Sixth International Conference on Advanced Computing(ICoAC) IEEE,2014.
[11] Luong Anh Tuan Nguyen†, Ba Lam To† ,Huu Khuong Nguyen† and Minh Hoang Nguyen : Detecting Phishing Web sites: A Heuristic URLBased Approach: In The 2013 International Conference on Advanced Technologies for Communications (ATC'13).
[12] Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013.
[13] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey: An Assessment of Features Related to Phishing Websites using an Automated Technique:In The 7th International Conference for Internet Technology and Secured Transactions,IEEE,2012.

c. PROBLEM STATEMENT DEFINITION

A problem statement is a concise description of the problem or issues a project seeks to address. The problem statement identifies the current state, the desired future state and any gaps between the two. A problem statement is an important communication tool that can help ensure everyone working on a project

knows what the problem they need to address is and why the project is important.

3. IDEATION & PROPOSED SOLUTION

a. EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Empathy mapping is a simple workshop activity that can be done with stakeholders, marketing and sales, product development, or creative teams to build empathy for end users.

Empathy maps are most useful at the beginning of the design process after user research but before requirements and concepting.

The mapping process can help synthesize research observations and reveal deeper insights about a user's needs. (The maps are most effective when based on research data, but like provisional personas, can be built using knowledge from internal participants or using existing persons) It can help guide the construction of personas or serve as a bridge between personas and concept deliverables.

b. IDEATION & BRAINSTORMING

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity. Brainstorming is usually conducted by getting a group of people together to come

up with either general new ideas or ideas for solving a specific problem or dealing with a specific situation.

c. PROPOSED SYSTEM

INCLUDE CONTENT

d. PROBLEM SOLUTION FIT

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem

THE PROBLEM SOLUTION FIT FOR WEB PHISHING DETECTION:

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves

the system, what changes the information, and where data is stored.

5.2 SOLUTION AND TECHNICAL ARCHITECTURE

A solution architecture (SA) is an architectural description of a specific solution. SAs combine guidance from different enterprise architecture viewpoints (business, information and technical), as well as from the enterprise solution architecture (ESA).

Technical Architecture (TA) is **a form of IT architecture that is used to design computer systems**. It involves the development of a technical blueprint about the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.

5.3 USER STORIES

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer. The user stories for web phishing detection:

6. PROJECT PLANNING AND SCHEULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

6.2 SPRINT DELIVERY SCHEDULE

Sprint is one timeboxed iteration of a continuous development cycle. Within a Sprint, planned amount of work has to be completed by the team and made ready for review. The term is mainly used in Scrum Agile methodology but somewhat basic idea of Kanban continuous delivery is also essence of Sprint Scrum.

6.3 REPORT FROM JIRA

When JIRA sends either standard notifications or user invitations to a mail server, they are listed as phishing attempts rather than legitimate emails
The mail server is receiving a constant stream of concurrent multiple email notifications from the same sender which, in turn, triggers security measures on the server which handle these messages as phishing attempts.

Resolution

· Check the base url of JIRA to see if it is set as a direct ip address with port number.

· Example: **http://10.10.10.10:8080**

· Some email servers (such as Microsoft Outlook) will consider messages from non-DNS urls as phishing attempts. You can correct this behaviour by setting JIRA's base url to a url address such as **http://my-jira.com.**

· Sometimes when certain mail servers receive multiple emails from the same sender security measures are triggered that will then list those emails as phishing messages. For this, it is best to check with the local mail server administrator for further assistance and confirmation.

7. CODING AND SOLUTIONING

a. HTML (Interface)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="This website is develop for identify the safety of url.">
    <meta name="keywords" content="phishing url,phishing,cyber security,machine learning,classifier,python">
    <meta name="author" content="VAIBHAV BICHAVE">

    <!-- BootStrap -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">
    <link href="static/styles.css" rel="stylesheet">
    <title>URL detection</title>
</head>

<body>
<div class=" container">
    <div class="row">
```

```
        <div class="form col-md" id="form1">
            <h2>PHISHING URL DETECTION</h2>
            <br>
            <form action="/" method ="post">
                <input type="text" class="form__input" name ='url' id="url"
placeholder="Enter URL" required="" />
                <label for="url" class="form__label">URL</label>
                <button class="button" role="button" >Check here</button>
            </form>
    </div>
    <div class="col-md" id="form2">
        <br>
        <h6 class = "right "><a href= {{ url }} target="_blank">{{ url
}}</a></h6>
        <br>
        <h3 id="prediction"></h3>
    </div>
</div>
<br>
<h1>GitHub Team ID : PNT2022TMID38146</h1>

</div>

    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+Or
CXaRkfj"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-
```

```
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvox
MfooAo"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh
/kR0JKI"
        crossorigin="anonymous"></script>

    <script>
            let x = '{{xx}}';
            let num = x*100;
            if (0<=x && x<0.50){
                num = 100-num;
            }
            let txtx = num.toString();
            if(x<=1 && x>=0.50){
                var label = "The URL is safe";
                document.getElementById("prediction").innerHTML = label;
                document.getElementById("button1").style.display="block";
            }
            else if (0<=x && x<0.50){
                var label = "The URL is not Safe"
                document.getElementById("prediction").innerHTML = label ;
                document.getElementById("button2").style.display="block";
            }
    </script>
</body>

</html>
```

b. App.py

```
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

file = open("pickle/model.pkl","rb")
gbc = pickle.load(file)
file.close()


app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)
        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
```

```
        return render_template('index.html',xx
=round(y_pro_non_phishing,2),url=url )
    return render_template("index.html", xx =-1)


if __name__ == "__main__":
    app.run(debug=True)
```

c. Feature.py

```
 import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

class FeatureExtraction:
    features = []
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""

        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(response.text, 'html.parser')
        except:
            pass

        try:
```

```
    self.urlparse = urlparse(url)
    self.domain = self.urlparse.netloc
except:
    pass

try:
    self.whois_response = whois.whois(self.domain)
except:
    pass




self.features.append(self.UsingIp())
self.features.append(self.longUrl())
self.features.append(self.shortUrl())
self.features.append(self.symbol())
self.features.append(self.redirecting())
self.features.append(self.prefixSuffix())
self.features.append(self.SubDomains())
self.features.append(self.Hppts())
self.features.append(self.DomainRegLen())
self.features.append(self.Favicon())


self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainURL())
self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags())
self.features.append(self.ServerFormHandler())
self.features.append(self.InfoEmail())
self.features.append(self.AbnormalURL())
self.features.append(self.WebsiteForwarding())
self.features.append(self.StatusBarCust())

self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
self.features.append(self.DNSRecording())
self.features.append(self.WebsiteTraffic())
self.features.append(self.PageRank())
self.features.append(self.GoogleIndex())
self.features.append(self.LinksPointingToPage())
```

```
        self.features.append(self.StatsReport())


    # 1.UsingIp
    def UsingIp(self):
        try:
            ipaddress.ip_address(self.url)
            return -1
        except:
            return 1

    # 2.longUrl
    def longUrl(self):
        if len(self.url) < 54:
            return 1
        if len(self.url) >= 54 and len(self.url) <= 75:
            return 0
        return -1

    # 3.shortUrl
    def shortUrl(self):
        match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                  'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                  'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                  'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr
\.im|link\.zip\.net', self.url)
        if match:
            return -1
        return 1

    # 4.Symbol@
    def symbol(self):
        if re.findall("@",self.url):
            return -1
        return 1

    # 5.Redirecting//
    def redirecting(self):
```

```
    if self.url.rfind('//')>6:
        return -1
    return 1

# 6.prefixSuffix
def prefixSuffix(self):
    try:
        match = re.findall('\-', self.domain)
        if match:
            return -1
        return 1
    except:
        return -1

# 7.SubDomains
def SubDomains(self):
    dot_count = len(re.findall("\.", self.url))
    if dot_count == 1:
        return 1
    elif dot_count == 2:
        return 0
    return -1

# 8.HTTPS
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
        return -1
    except:
        return 1

# 9.DomainRegLen
def DomainRegLen(self):
    try:
        expiration_date = self.whois_response.expiration_date
        creation_date = self.whois_response.creation_date
        try:
            if(len(expiration_date)):
                expiration_date = expiration_date[0]
        except:
            pass
        try:
            if(len(creation_date)):
```

```
                creation_date = creation_date[0]
            except:
                pass

            age = (expiration_date.year-creation_date.year)*12+ (expiration_date.month-
creation_date.month)
            if age >=12:
                return 1
            return -1
        except:
            return -1

    # 10. Favicon
    def Favicon(self):
        try:
            for head in self.soup.find_all('head'):
                for head.link in self.soup.find_all('link', href=True):
                    dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                    if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:
                        return 1
            return -1
        except:
            return -1

    # 11. NonStdPort
    def NonStdPort(self):
        try:
            port = self.domain.split(":")
            if len(port)>1:
                return -1
            return 1
        except:
            return -1

    # 12. HTTPSDomainURL
    def HTTPSDomainURL(self):
        try:
            if 'https' in self.domain:
                return -1
            return 1
        except:
            return -1

    # 13. RequestURL
    def RequestURL(self):
```

```
        try:
            for img in self.soup.find_all('img', src=True):
                dots = [x.start(0) for x in re.finditer('\.', img['src'])]
                if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for audio in self.soup.find_all('audio', src=True):
                dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
                if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for embed in self.soup.find_all('embed', src=True):
                dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
                if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for iframe in self.soup.find_all('iframe', src=True):
                dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
                if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            try:
                percentage = success/float(i) * 100
                if percentage < 22.0:
                    return 1
                elif((percentage >= 22.0) and (percentage < 61.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1

    # 14. AnchorURL
    def AnchorURL(self):
        try:
            i,unsafe = 0,0
            for a in self.soup.find_all('a', href=True):
                if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not (url in
a['href'] or self.domain in a['href']):
```

```
            unsafe = unsafe + 1
        i = i + 1

    try:
        percentage = unsafe / float(i) * 100
        if percentage < 31.0:
            return 1
        elif ((percentage >= 31.0) and (percentage < 67.0)):
            return 0
        else:
            return -1
    except:
        return -1

except:
    return -1

# 15. LinksInScriptTags
def LinksInScriptTags(self):
  try:
    i,success = 0,0

    for link in self.soup.find_all('link', href=True):
        dots = [x.start(0) for x in re.finditer('\.', link['href'])]
        if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:
            success = success + 1
        i = i+1

    for script in self.soup.find_all('script', src=True):
        dots = [x.start(0) for x in re.finditer('\.', script['src'])]
        if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
            success = success + 1
        i = i+1

    try:
        percentage = success / float(i) * 100
        if percentage < 17.0:
            return 1
        elif((percentage >= 17.0) and (percentage < 81.0)):
            return 0
        else:
            return -1
    except:
        return 0
  except:
```

```
        return -1

    # 16. ServerFormHandler
    def ServerFormHandler(self):
        try:
            if len(self.soup.find_all('form', action=True))==0:
                return 1
            else :
                for form in self.soup.find_all('form', action=True):
                    if form['action'] == "" or form['action'] == "about:blank":
                        return -1
                    elif self.url not in form['action'] and self.domain not in form['action']:
                        return 0
                    else:
                        return 1
        except:
            return -1

    # 17. InfoEmail
    def InfoEmail(self):
        try:
            if re.findall(r"[mail\(\)|mailto:?]", self.soap):
                return -1
            else:
                return 1
        except:
            return -1

    # 18. AbnormalURL
    def AbnormalURL(self):
        try:
            if self.response.text == self.whois_response:
                return 1
            else:
                return -1
        except:
            return -1

    # 19. WebsiteForwarding
    def WebsiteForwarding(self):
        try:
            if len(self.response.history) <= 1:
                return 1
            elif len(self.response.history) <= 4:
                return 0
```

```
        else:
            return -1
    except:
        return -1

# 20. StatusBarCust
def StatusBarCust(self):
    try:
        if re.findall("<script>.+onmouseover.+</script>", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

# 21. DisableRightClick
def DisableRightClick(self):
    try:
        if re.findall(r"event.button ?== ?2", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

# 22. UsingPopupWindow
def UsingPopupWindow(self):
    try:
        if re.findall(r"alert\(", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

# 23. IframeRedirection
def IframeRedirection(self):
    try:
        if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
```

```
    # 24. AgeofDomain
    def AgeofDomain(self):
        try:
            creation_date = self.whois_response.creation_date
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass

            today  = date.today()
            age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
            if age >=6:
                return 1
            return -1
        except:
            return -1

    # 25. DNSRecording
    def DNSRecording(self):
        try:
            creation_date = self.whois_response.creation_date
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass

            today  = date.today()
            age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
            if age >=6:
                return 1
            return -1
        except:
            return -1

    # 26. WebsiteTraffic
    def WebsiteTraffic(self):
        try:
            rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="
+ url).read(), "xml").find("REACH")['RANK']
            if (int(rank) < 100000):
                return 1
            return 0
        except :
```

```
            return -1

    # 27. PageRank
    def PageRank(self):
        try:
            prank_checker_response = requests.post("https://www.checkpagerank.net/index.php",
{"name": self.domain})

            global_rank = int(re.findall(r"Global Rank: ([0-9]+)", rank_checker_response.text)[0])
            if global_rank > 0 and global_rank < 100000:
                return 1
            return -1
        except:
            return -1


    # 28. GoogleIndex
    def GoogleIndex(self):
        try:
            site = search(self.url, 5)
            if site:
                return 1
            else:
                return -1
        except:
            return 1

    # 29. LinksPointingToPage
    def LinksPointingToPage(self):
        try:
            number_of_links = len(re.findall(r"<a href=", self.response.text))
            if number_of_links == 0:
                return 1
            elif number_of_links <= 2:
                return 0
            else:
                return -1
        except:
            return -1

    # 30. StatsReport
    def StatsReport(self):
        try:
            url_match = re.search(
```

```
'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly', url)
            ip_address = socket.gethostbyname(self.domain)
            ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'

'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'

'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'

'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'

'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|'

'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
            if url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1

    def getFeaturesList(self):
        return self.features
```

7.1 FEATURE 1

The webpage designed on the basis of a very user-friendly interface design which is easy to use. The working of the Web application is very simple and is at a beginner basis considering the user only has to apply the required URL to the search box and click Check which automatically checks for all the information that is to be displayed to the user about the website.

The process is fully automated and does require any help from the user expect for them to provide the webpage with the URL to be tested and the rest of the process done by the system. The URL is tested through various datasets that determines whether the webpage already exists with the Dataset.

7.2 FEATURE 2

URL shortening is a technique on the World Wide Web in which a Uniform Resource Locator (URL) may be made substantially shorter and still direct to the required page. This is achieved by using a redirect which links to the web page that has a long URL.

For example: the URL
"https://example.com/assets/category_B/subcategory_C/Foo/" can be shortened to “https://example.com/Foo",
and the URL
"https://en.wikipedia.org/wiki/URL shortening"
can be shortened to "https://w.wiki/U". Often the redirect domain name is shorter than the original one.

The shortened URL can also be checked by using this Webpage Application which provided will also function perfectly, with this type of URL’s that commonly does not work on other sites.

Third Party Datasets are updated regularly.

The accuracy rate of detection of these phishing websites are 99%.

7.2 DATABASE SCHEMA

a. Provided database

IBM Dataset

b. Thirdparty database

https://phishtank.com/api/v1/

8. TESTING

8.1 TEST CASES

a. **Test case 1 :** User Interface

b. **Test case 2 :** Prediction model

c. **Test case 3** : Classification

d. **Test case 4 :** Third party API

e. **Test case 5 :** Pre-processed data.

8.2 USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.
The main **Purpose of UAT** is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.
UAT is performed by –
. Client

· End users

UAT PROCESS

Need of User Acceptance Testing arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed

9. RESULTS

9.1 PERFORMANCE METRICS

In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as Yes or No, 0 or 1, Spam or Not Spam, etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

· Accuracy
· Confusion Matrix
· Precision
· Recall
· F-Score
· AUC(Area Under the Curve)-ROC

10. ADVANTAGES:

1. This system can be used by many E- commerce or other websites in order to have good customer relationship.
2. User can make online payment securely.
3. With the help of this system user can also purchase products online without any hesitation.
4. KNN algorithm and decision tree algorithm used in this system provides better performance.

DISADVANTAGES:

1. If internet connection fails, this system won't work.
2. All websites related data will be stored in one place.

11. CONCLUSION

Currently phishing attacks are so common because it can attack globally and capture and store the users' confidential information. This information is used by the attackers which are indirectly involved in the phishing process. The main objective of this paper is to provide the protection from two kinds of phishing attacks. Hence by using the Password hashing technique by MD5 algorithm, the proposed system fulfilled its needs.

The proposed system only confined to the single bank that is Indian bank. The phisher can’t do anything without the session key, but it is not so hard to get the session key from the authorized users mobile with the help of hacker.

12. FUTURE SCOPE

As a future work, the proposed system will be expanding to other banks also and the system has to provide the security not only against the phishing but also the hacking.

13.APPENDIX

SOURCE CODE

app.py

```
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

file = open("pickle/model.pkl","rb")
gbc = pickle.load(file)
file.close()


app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
```

```
def index():
    if request.method == "POST":
        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)
        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
         return render_template('index.html',xx =round(y_pro_non_phishing,2),url=url
)
    return render_template("index.html", xx =-1)


if __name__ == "__main__":
    app.run(debug=True)
```

GITHUB AND PROJECT DEMO LINK

Github : https://github.com/IBM-EPBL/IBM-Project-5284-1658755640

Demonstration video : https://youtu.be/ngyAokidw7M