A Project Report

On

PHISHING DETECTION USING HADOOP- MAP REDUCE AND MACHINE LEARNING

Submitted in partial fulfilment of the requirement of

University of Mumbai

For the Degree of

Bachelor of Engineering

in

COMPUTER ENGINEERING

Submitted by

A.ANNIE GRACE
TANYA SERAH JACOB
SWAPNIL MONTEIRO

Supervised by

MR. MRITUNJAY OJHA

Department of Computer Engineering

Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai - 400703

UNIVERSITY OF MUMBAI

2019-2020

APPROVAL SHEET

This is to certify that the project entitled

"PHISHING DETECTION USING HADOOP-MAP REDUCE AND MACHINE LEARNING"

Submitted by

A.ANNIE GRACE	**101601**
TANYA SERAH JACOB	**101656**
SWAPNIL MONTEIRO	**101670**

Supervisors :	
Project Coordinator :	
Examiners :	**1.**
	2.
Head of Department :	

Date :

Place :

Declaration

We declare that this written submission for B.E. Declaration entitled "**Phishing detection using hadoop- map reduce and machine learning**" represent our ideas in our own words and where others' ideas or words have been included. We have adequately cited and referenced the original sources. We also declared that we have adhere to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by institute and also evoke penal action from the sources which have thus not been properly cited or from whom paper permission have not been taken when needed.

Project Group Members:

1. A. Annie Grace, 101601

2. Tanya Serah Jacob, 101656

3. Swapnil Monteiro, 101670

Abstract

Today phishing attacks have become one of the most serious issues that internet users, organisations and service providers face. In phishing attack, attacker attempts to acquire the user's personal information by using spoofed addresses, bogus websites or both. The user will be the target of such activities because phishing web pages look very similar to real ones, so it's hard to distinguish between ham websites and genuine ones because it's very difficult to detect this kind of webpage because we need to remember certain attributes that inexperienced users might not know for identification. So Machine Learning and Hadoop-Map Reduce are used in this proposed system.Machine learning and Hadoop-Map Reduce are used in this proposed system to quickly retrieve URL attributes, which play a key role in recognizing phishing web pages and it is known for its time efficiency.Throughput is also gained using this.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Phishing is a fraudulent attempt, usually made through email, to steal your personal information. Phishing websites are forged websites created by malicious people to mimic real websites. The victims of these phishing websites or emails might expose their personal information like the passwords of their credit card, bank account etc. This normally results into the financial loss of the victim. Currently the systems available in the market like anti- phishing toolbars , anti-phishing filters [2] embedded in the antivirus programs, etc. rely mainly on the database obtained by checking the suspicious websites amp; storing the phishy websites in the database. The procedure involves a number of steps. Firstly the anti-phishing program tries to detect the phishing website on its accord, if it failed to decide about the authenticity of the program, it simply sends the URL to the server for checking the authenticity. The testers check whether the site is original or not, if not, then an entry is made in the database amp; the updated database is sent to all the known users of the anti-phishing product. The whole process is lengthy, so relying totally on it might prove dangerous. The solution is to check the website attributes, as many possible, in the real time environment. The less the suspicious values of the attributes, more the authenticity of the website, and vice versa. Since the process is done in the real time, the user doesn't need to wait till the update is made amp; downloaded in proposed system. User is safe from the phishing websites if user uses our proposed system.

1.2 Motivation

The motivation and challenges of the project are:

1- There are shortcomings in the current anti-phishing techniques and solutions which allow some sophisticated attackers to achieve what the want.

2- The number of victims, whether an individual Phishing Detection Using Hadoop and Machine Learning or an organisation, has increased over the last couple of years. For instance, in 2008, more than 4 million US internet users lost major amounts of their money.

3- Phishing attackers mainly try to gather users confidential data and use that in a malicious way. This system will try to reduce these attacks by detecting the fake or phished websites and alerting the users from those websites

1.3 Aim and Objective

Our aim is to let user know about phishing websites to stop attacker from stealing personal information of the user.

Our Objectives are:

1.We detect the phishing websites.

2.It evokes a warning sign.

3.If the site is non phishing then it takes the user to that legitimate website.

Chapter 2

Study Of the System

2.1 About the Technique

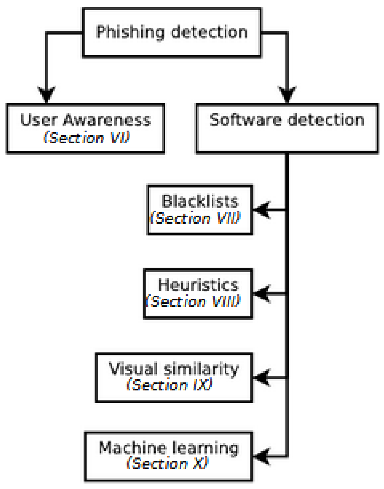

Phishing detection
User Awareness (Section VI)
Software detection
Blacklists (Section VII)
Heuristics (Section VIII)
Visual similarity (Section IX)
Machine learning (Section X)

Figure 2.1: classification of phishing detection

The term phishing comes from fishing in a sense that fishers (i.e. attackers) use a bait (i.e. socially-engineered messages)to fish (e.g. steal personal information of victims) The Traditional method includes by learning from previous phishing campaigns, it is possible to enhance the detection of future phishing campaigns. Such learning can be performed by a human observer, or software.
For user level security toolbars are developed as add-on to the browsers. Netcraft toolbar for the Mozilla browser.
The other way is to use of Black Lists by the browsers. Black List is maintained by browsers like Google Crome (Google Safe Browsing). These Black Lists are updated to time manually by hiring expert who manually categories suspected URL whether they are Genuine of phishing.
Anti-phishing working Group (AWPG) helps its partners to build anti-phishing solutions. AWPG generates monthly reports about the current phishing activities. It keeps an eye on phishing activities all over the world and collaborates with its partners with the information obtained.

2.2 Various Available Technique

1.Using Machine Learning
2.Using Big Data
3.Using Map Reduce and Part Algorithm
4.Using Text analysis and link analysis
5.Using Fuzzy techniques

2.3 Related Works

2.3.1 Phishing Website Detection using Machine Learning : A Review

Phishing frauds might be the most popular cyber crime used today. There are various domains where phishing attack can occur like online payment sector, web mail, and financial institution, file hosting or cloud storage and many others. The web mail and online payment sector was targeted by phishing more than in any other industry sector. Several anti-phishing techniques are there such as blacklist, heuristic, visual similarity and machine learning. From this, blacklist approach is commonly used because it is easy to use and implement but it fails to detect new phishing attacks. Machine Learning is efficient technique to detect phishing. It also removes drawback of existing approach. We perform detailed literature survey and proposed new approach to detect phishing website by features extraction and machine learning algorithm.

2.3.2 Anti-phishing using Big Data

The users will be victim for this kind of activities, because phishing web pages looks very similar to real ones, so finds difficult to distinguish between the fake website and ones, detecting this kind of webpage is very difficult because for identification it takes several attributes into consideration which user might not knowing those things. The existing phishing detection systems are highly dependent on database and they are very time consuming also.
In this paper, Hadoop-Map Reduce is used for fast retrieval of URL attributes, which plays a key role in identifying phishing web pages and it is known for its time efficiency and throughput also gained using this.The existing phishing detection systems are highly dependent on database and they are very time consuming.

2.3.3 Analysis and detection of email phishing using pyspark

Phishing is one of the luring techniques used by phishing artist in the intention of exploiting the personal details of unsuspected users. Phishing website is a mock website that looks similar in appearance but different in destination. The unsuspected users post their data thinking that these websites come from trusted financial institutions. Big data refers to an enormous amount of data set that is able to expose patterns associated with human interaction through computational analysis. The main purpose here is to detect the e-mails which user receives is legitimate or not. The goals of our paper as well as system are: (a) To provide security (b) Accuracy in detection

2.3.4 An approach for Malicious Spam Detection In Email with comparison of different classifiers.

Today, one of the cheapest form of communication in the world is email, and its simplicity makes it vulnerable to many threats. One of the most important threats to email is spam; unsolicited email, especially when advertising agency send a mass mail. Spam email may also include malware as scripts or other executable file. Sometimes they also consist harmful attachments or links to phishing websites. This malicious spam threatens the privacy and security of large amount of sensitive data. Hence, a system that can automatically learn how to classify malicious spam in email is highly desirable. In this paper, we aim to improve detection of malicious spam through feature selection. We propose a model that employs a novel dataset for the process of feature selection, a step for improving classification in later stage. Feature selection is expected to improve training time and accuracy of malicious spam detection. This paper also shows the comparison of various classifier used during the process

Chapter 3

Proposed System

3.1 Problem Statement

Now a day's phishing attack has become one of the most serious issues faced by internet users, organizations and service providers. In phishing attack attacker tries to obtain the personal information of the users by using spoofed emails or by using fake websites or both. The users will be victim for this kind of activities, because phishing web pages looks very similar to real ones, so finds difficult to distinguish between the fake website and ones, detecting this kind of webpage is very difficult because for identification it takes several attributes into consideration which user might not knowing those things. In this proposed system, Machine learning and Hadoop-Map Reduce is used for fast retrieval of URL attributes, which plays a key role in identifying phishing web pages and it is known for its time efficiency and throughput also gained using this.

3.2 Scope

Main goal of the system is to achieve speed up in existing anti-phishing system by some means. Using Hadoop- MapReduce in integration with anti-phishing technique we have achieved considerable time speedup. Even if the phishing webpage is not showing phishing characteristics very clearly in the first layer it might show characteristics in the next layer so that no phishing website will pass through our system. This is the advantage of having layered architecture of attributes. Hadoop MapReduce will increase the response time of the system considerably. This system is very effective in securing network from phishing attack even at its best.

3.3 Proposed System

The user will enter the URL of the webpage, she wishes to visit. Using that URL, we will download the source code of the webpage then decide the values of the attributes.For finding these values we will make use of Hadoop Map Reduce [9].This will speed up the process of attribute value assignment.Basic word count example [10] of Hadoop-MapReduce is used to search sensitive words in webpages. In same way wherever required help of Hadoop is taken. These calculated attributes are the input to the Prediction module. Based on the records stored from www.phishtank.com database, training data is prepared. All the characteristics of reported phishing website at phishtank.com corpus are studied and based on that attributes are decided and training data for machine learning algorithm is prepared. Using training data machine learning algorithm generates set of rules based on which decision is to be made. Prediction module gets two inputs rules generated by machine learning algorithm and attribute found from requested URL. Prediction module finally predict URL falls under which category (Phishing, Legitimate, and Doubtful).

Chapter 4

Design Of the System

4.1 Requirement Engineering

4.1.1 Requirement Elicitation

The Model we have made are been updated automatically. Everytime new url are being fed into the system. The system learns according to supervised learning in machine Learning. The user enters the url and its been checked whether its phished or not and then its been fed into the database, also its present in history so the the user can log in to see which all urls are been phished.

4.1.2 UML diagrams

4.1.2.1 Use Case Diagram

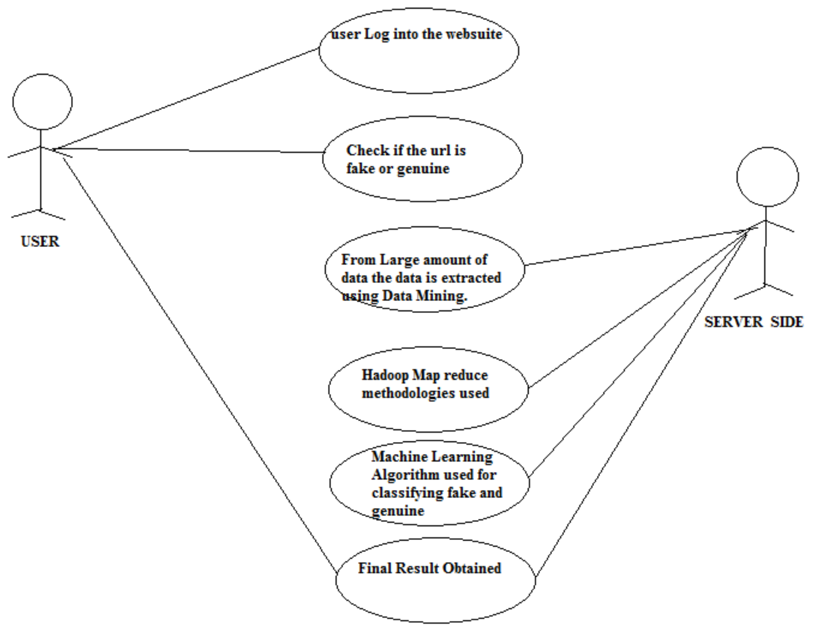

user Log into the websuite
Check if the url is fake or genuine
USER
From Large amount of data the data is extracted using Data Mining.
SERVER SIDE
Hadoop Map reduce methodologies used
Machine Learning Algorithm used for classifying fake and genuine
Final Result Obtained

Figure 4.1: Use Case Diagram

The Use case diagram shows that the operations which the user performs and how the proposed system works

4.1.2.2 Sequence Diagram

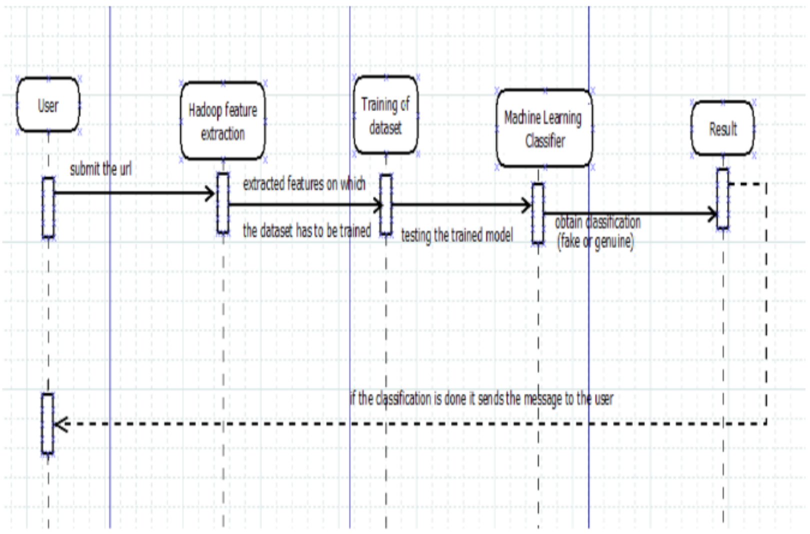

User
Hadoop feature extraction
Training of dataset
Machine Learning Classifier
Result
submit the url
extracted features on which the dataset has to be trained
testing the trained model
obtain classification (fake or genuine)
if the classification is done it sends the message to the user

Figure 4.2: Sequence Diagram

The sequence diagram shows the flow of various operations within the proposed system.

4.1.2.3 Activity Diagram

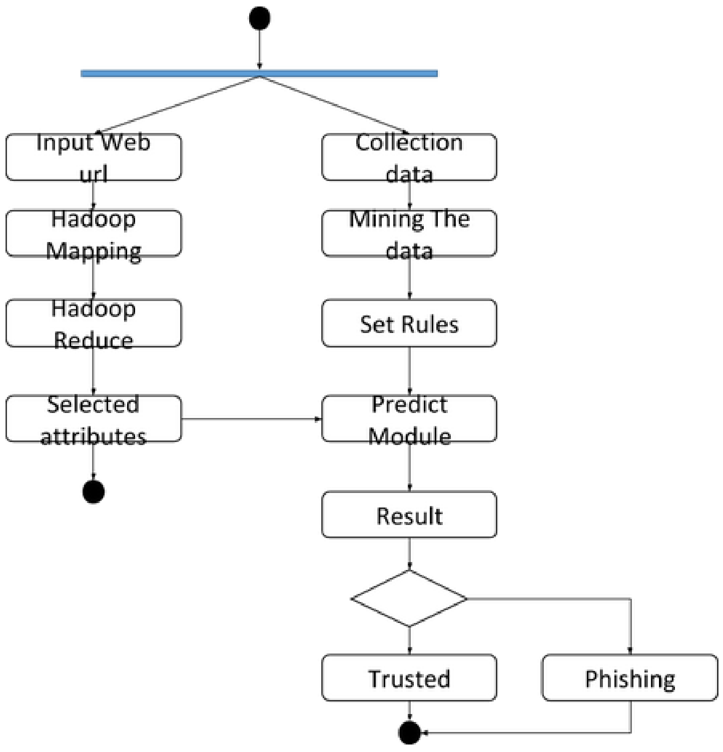


Figure 4.3: Activity Diagram

The Activity diagram shows the step by step activities performed after the url being entered into the the system.

4.1.2.4 Software Requirements

O/S : Windows XP.
Language : Java.
IDE : Net Beans 6.9.1
Data Base : My Sql

4.1.2.5 Hardware Requirements

System : Pentium IV 2.4 GHz
Hard Disk : 160 GB
Monitor : 15 VGA color
Mouse : Lenovo
Keyboard : 110 keys enhanced
Ram : 2GB

4.1.2.6 Modules Required

subsubsectionProcess Dataset

• The user will enter the URL of the webpage, she wishes to visit. Using that URL, we will download the source code of the webpage amp; then decide the values of the attributes.

• This Module Performs to get the web site URL and get the data set regarding to our system.

• Process the dataset and add the data set into database tables.

• Preprocessing the dataset for adopt dataset.

4.2 System Architecture

4.2.1 Block Diagram

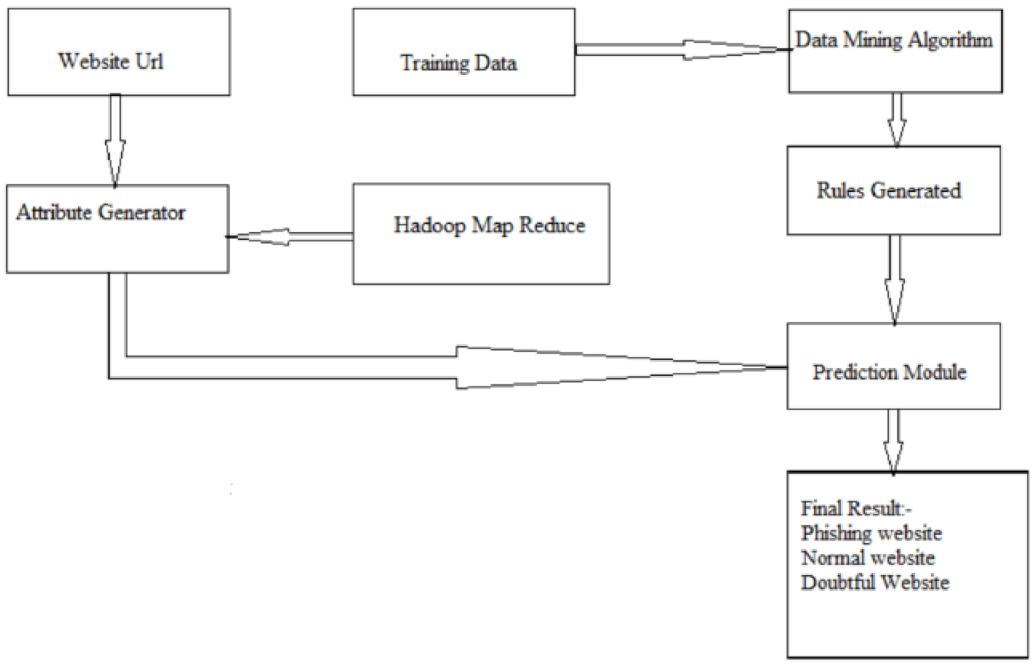


Figure 4.4: Block Diagram

Chapter 5

Result and Discussion

5.1 Screenshots of the System

5.1.1 Phishing Detection System

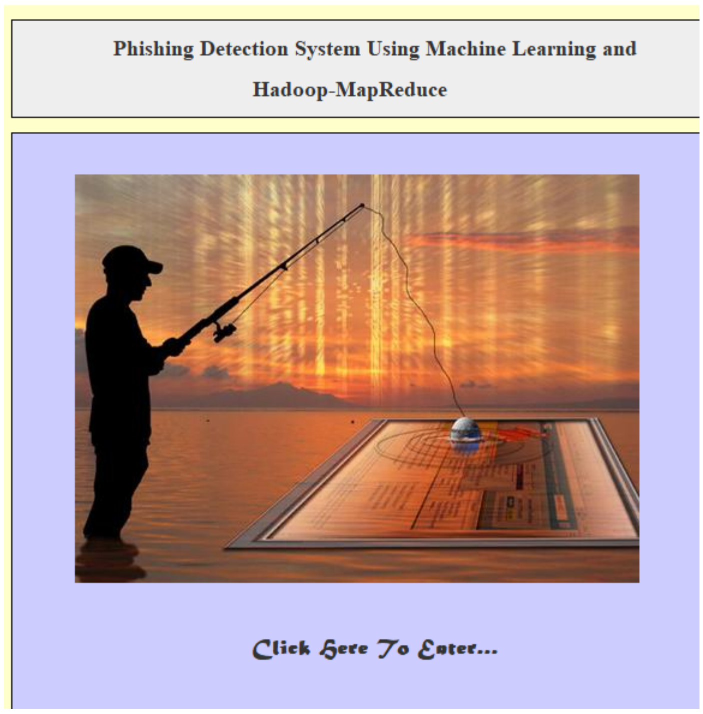
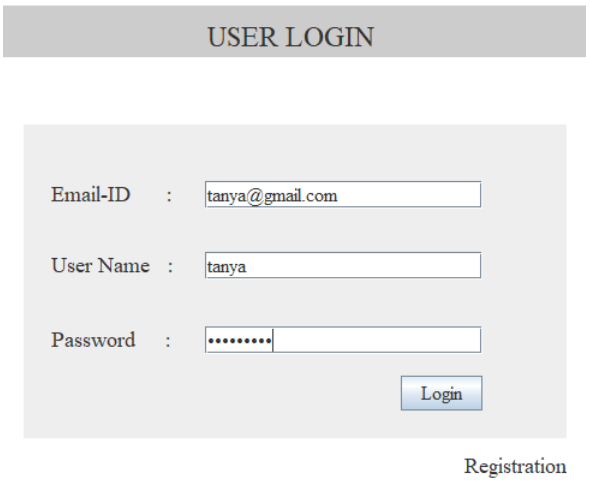
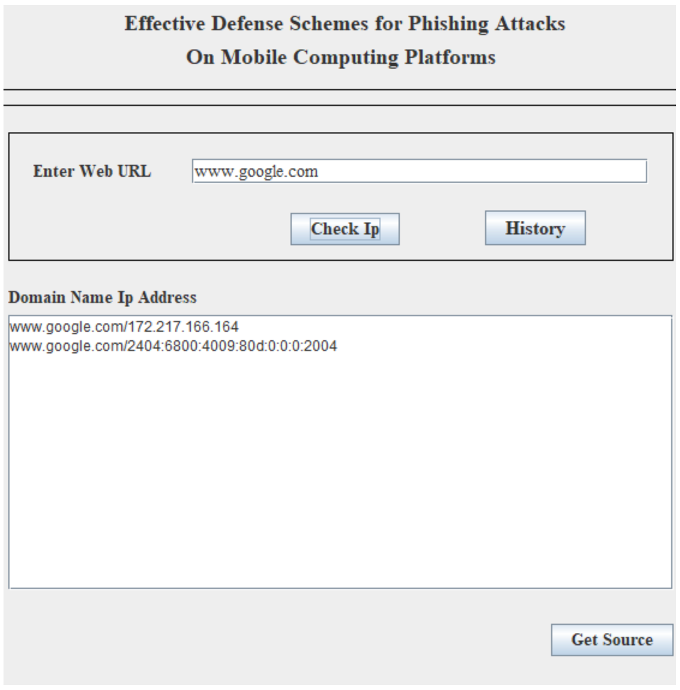

Phishing Detection System Using Machine Learning and Hadoop-MapReduce

Figure 5.1: Home Page


USER LOGIN

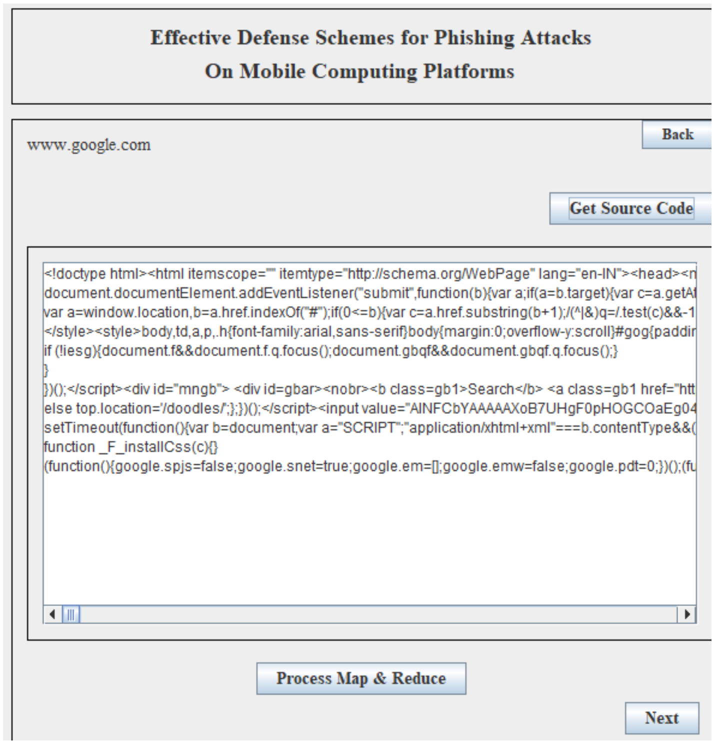
Email-ID	:	tanya@gmail.com
User Name	:	tanya
Password	:	•••••••••

Figure 5.2: Login Page


Effective Defense Schemes for Phishing Attacks
On Mobile Computing Platforms
Enter Web URL
www.google.com
Check Ip
History
Domain Name Ip Address
Get Source

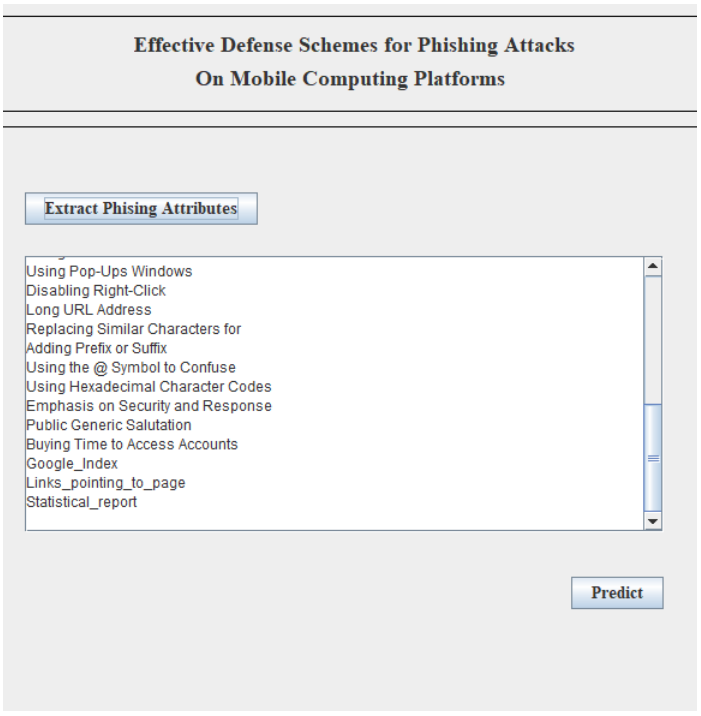
```
www.google.com/172.217.166.164
www.google.com/2404:6800:4009:80d:0:0:0:2004
```

Figure 5.3: Check url/ check ip page


Effective Defense Schemes for Phishing Attacks
On Mobile Computing Platforms
www.google.com
Back
Get Source Code
Process Map & Reduce
Next

```
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-IN"><head><n
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=a.getAt
var a=window.location,b=a.href.indexOf("#");if(0<=b){var c=a.href.substring(b+1);/(^|&)q=/.test(c)&&-1
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overflow-y:scroll}#gog{paddin
if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}
}
})();</script><div id="mngb"> <div id=gbar><nobr><b class=gb1>Search</b> <a class=gb1 href="htt
else top.location='/doodles/';};})();</script><input value="AINFCbYAAAAAXoB7UHgF0pHOGCOaEg04
setTimeout(function(){var b=document;var a="SCRIPT";"application/xhtml+xml"===b.contentType&&(
function _F_installCss(c){}
(function(){google.spjs=false;google.snet=true;google.em=[];google.emw=false;google.pdt=0;})();(fu
```

Figure 5.4: Collecting source code


Effective Defense Schemes for Phishing Attacks
On Mobile Computing Platforms
Extract Phising Attributes
Predict

Using Pop-Ups Windows
Disabling Right-Click
Long URL Address
Replacing Similar Characters for
Adding Prefix or Suffix
Using the @ Symbol to Confuse
Using Hexadecimal Character Codes
Emphasis on Security and Response
Public Generic Salutation
Buying Time to Access Accounts
Google_Index
Links_pointing_to_page
Statistical_report

Figure 5.5: Extracting Features

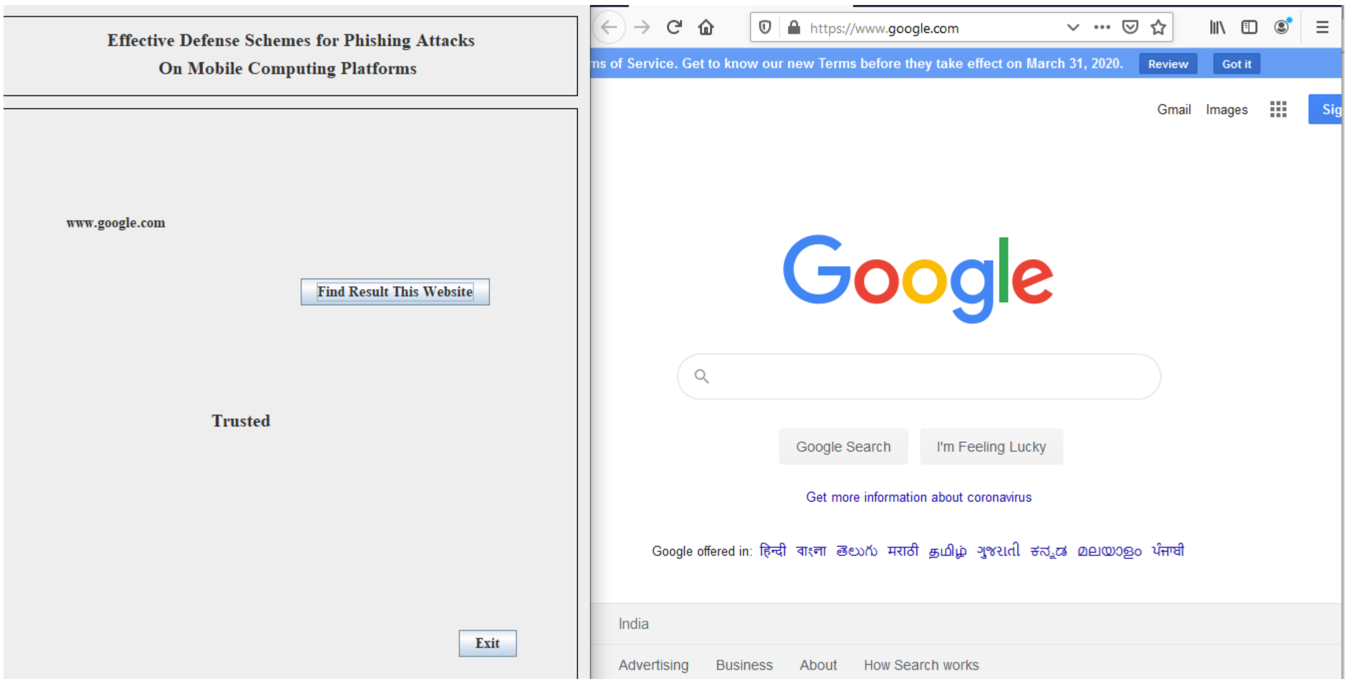

Effective Defense Schemes for Phishing Attacks
On Mobile Computing Platforms
www.google.com
Find Result This Website
Trusted
Exit
https://www.google.com
ns of Service. Get to know our new Terms before they take effect on March 31, 2020.
Review
Got it
Gmail
Images
Sig
Google
Google Search
I'm Feeling Lucky
Get more information about coronavirus
Google offered in: हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ
India
Advertising
Business
About
How Search works

Figure 5.6: Prediction

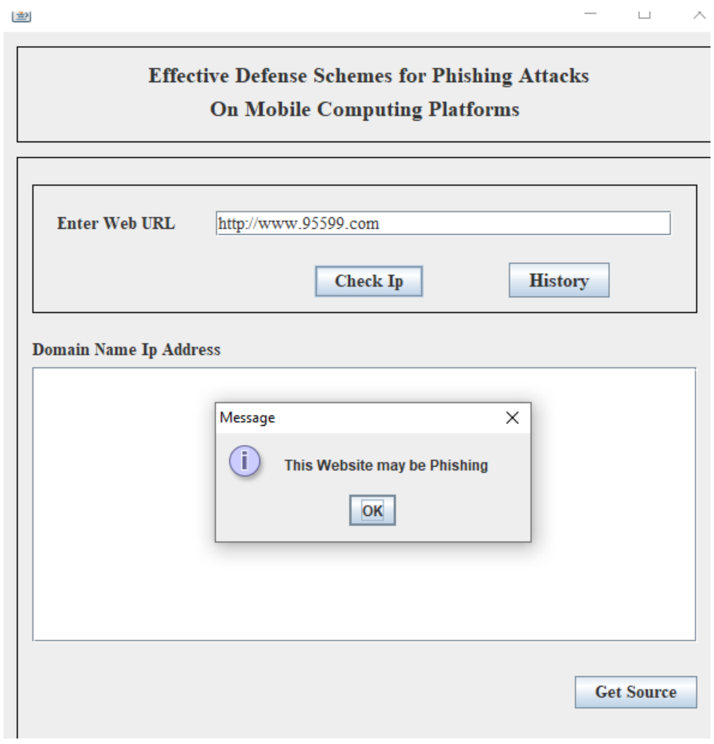

Effective Defense Schemes for Phishing Attacks
On Mobile Computing Platforms
Enter Web URL
http://www.95599.com
Check Ip
History
Domain Name Ip Address
Message
This Website may be Phishing
OK
Get Source

Figure 5.7: For Fake URL

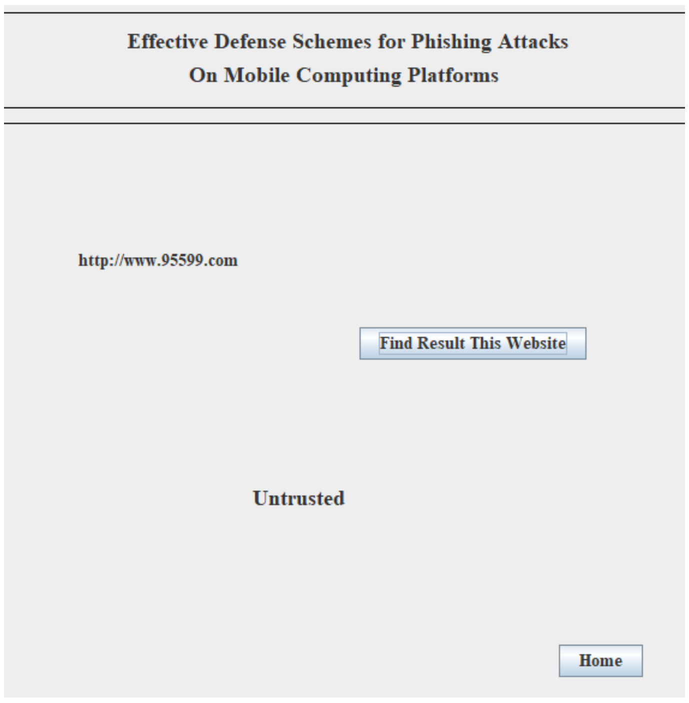
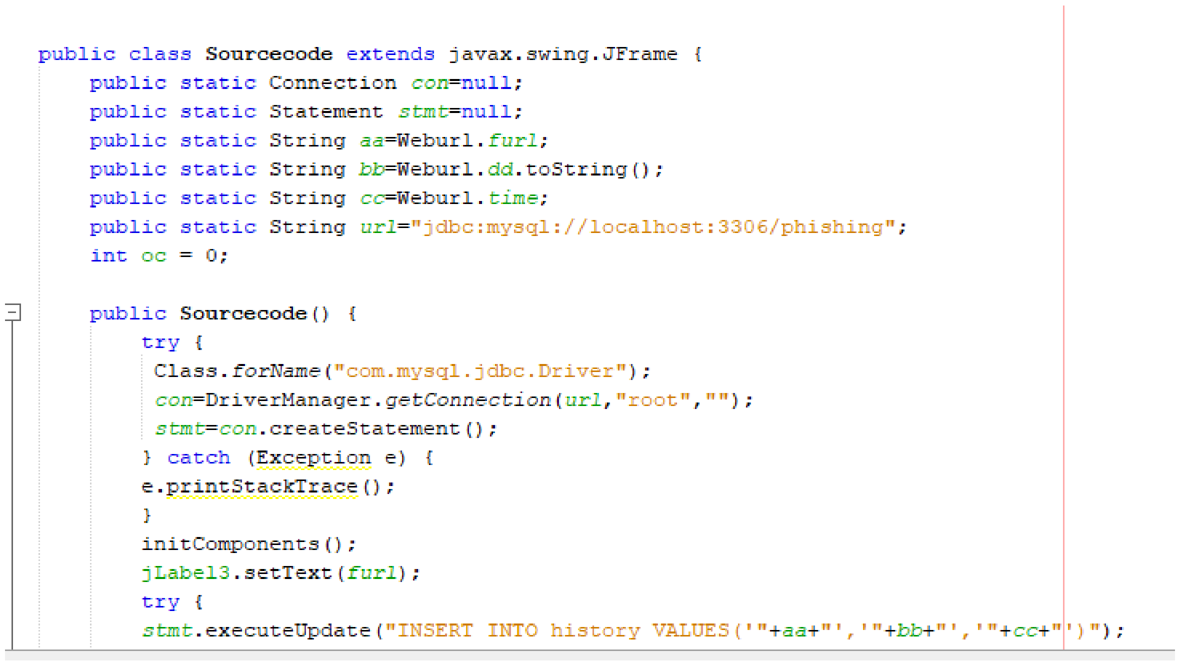

Effective Defense Schemes for Phishing Attacks
On Mobile Computing Platforms
http://www.95599.com
Find Result This Website
Untrusted
Home

Figure 5.8: Output as Untrusted

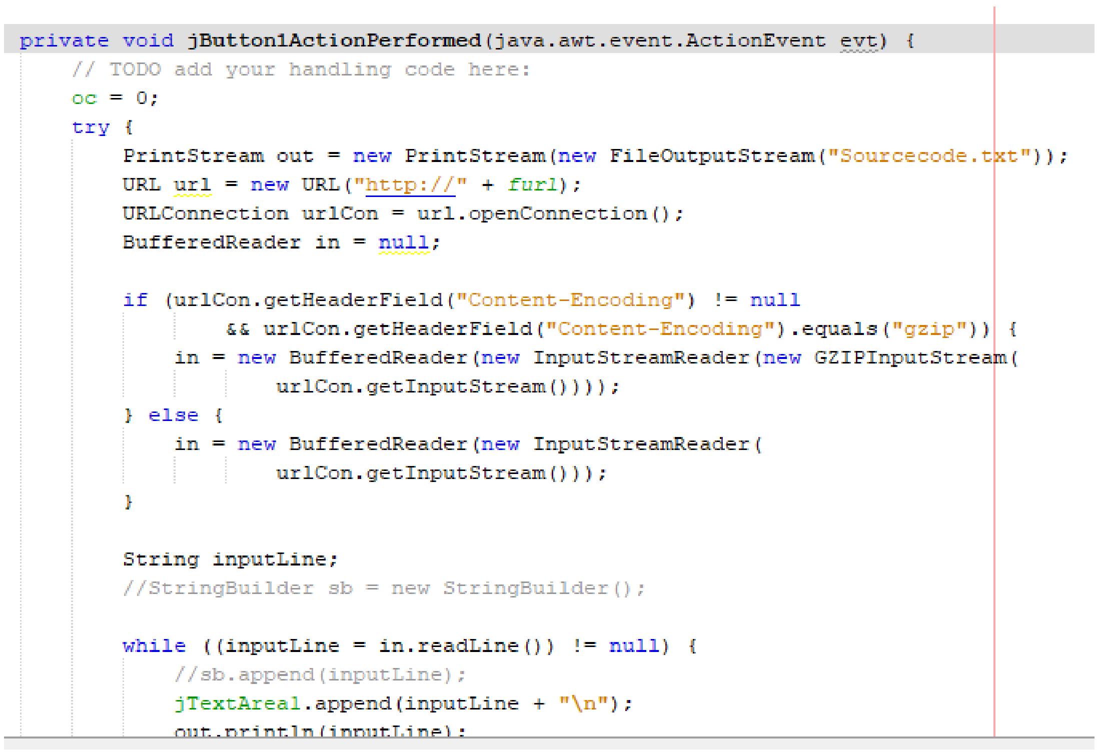
5.2 Sample Code (of imp part/ main logic)

5.2.1 Sourcecode Extraction code



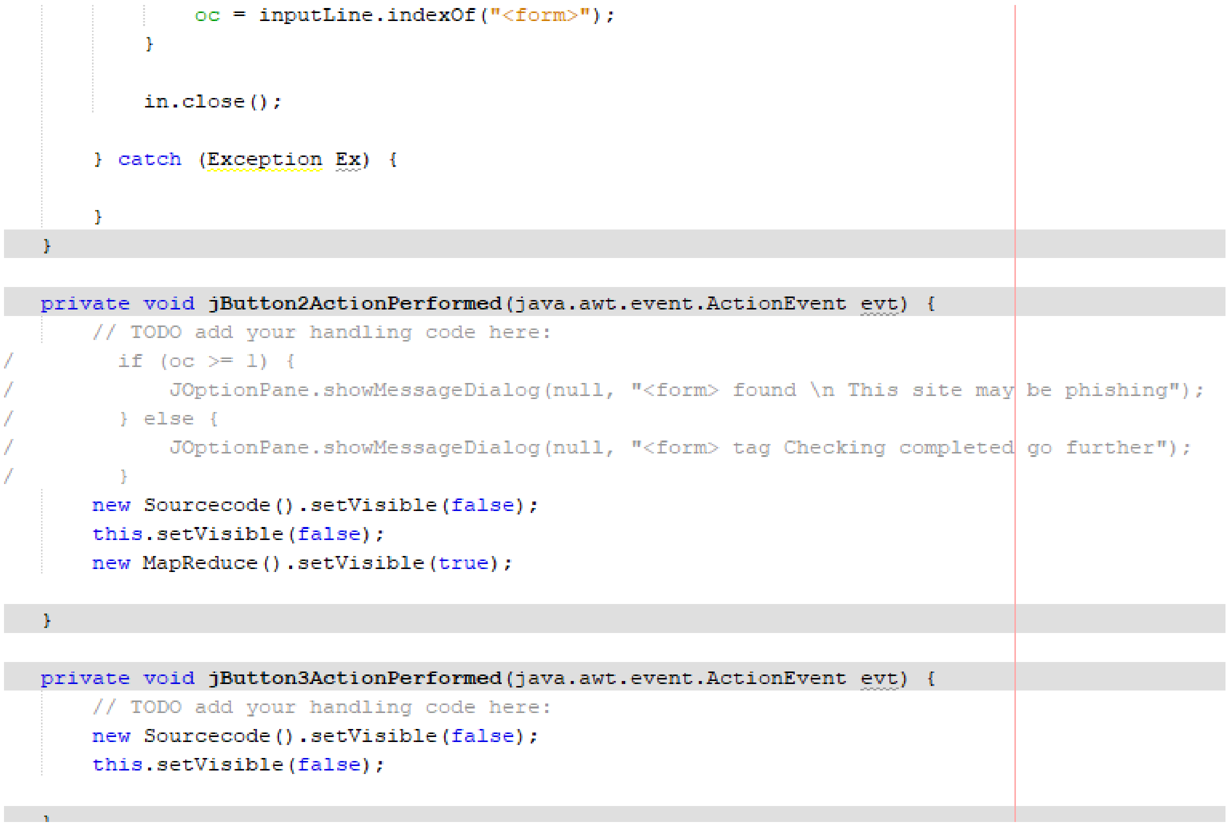
```
public class Sourcecode extends javax.swing.JFrame {
    public static Connection con=null;
    public static Statement stmt=null;
    public static String aa=Weburl.furl;
    public static String bb=Weburl.dd.toString();
    public static String cc=Weburl.time;
    public static String url="jdbc:mysql://localhost:3306/phishing";
    int oc = 0;

    public Sourcecode() {
        try {
         Class.forName("com.mysql.jdbc.Driver");
         con=DriverManager.getConnection(url,"root","");
         stmt=con.createStatement();
        } catch (Exception e) {
        e.printStackTrace();
        }
        initComponents();
        jLabel3.setText(furl);
        try {
        stmt.executeUpdate("INSERT INTO history VALUES('"+aa+"','"+bb+"','"+cc+"')");
```

Figure 5.9: Extraction of source code of url



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    oc = 0;
    try {
        PrintStream out = new PrintStream(new FileOutputStream("Sourcecode.txt"));
        URL url = new URL("http://" + furl);
        URLConnection urlCon = url.openConnection();
        BufferedReader in = null;

        if (urlCon.getHeaderField("Content-Encoding") != null
                && urlCon.getHeaderField("Content-Encoding").equals("gzip")) {
            in = new BufferedReader(new InputStreamReader(new GZIPInputStream(
                    urlCon.getInputStream())));
        } else {
            in = new BufferedReader(new InputStreamReader(
                    urlCon.getInputStream()));
        }

        String inputLine;
        //StringBuilder sb = new StringBuilder();

        while ((inputLine = in.readLine()) != null) {
            //sb.append(inputLine);
            jTextArea1.append(inputLine + "\n");
            out.println(inputLine);
```

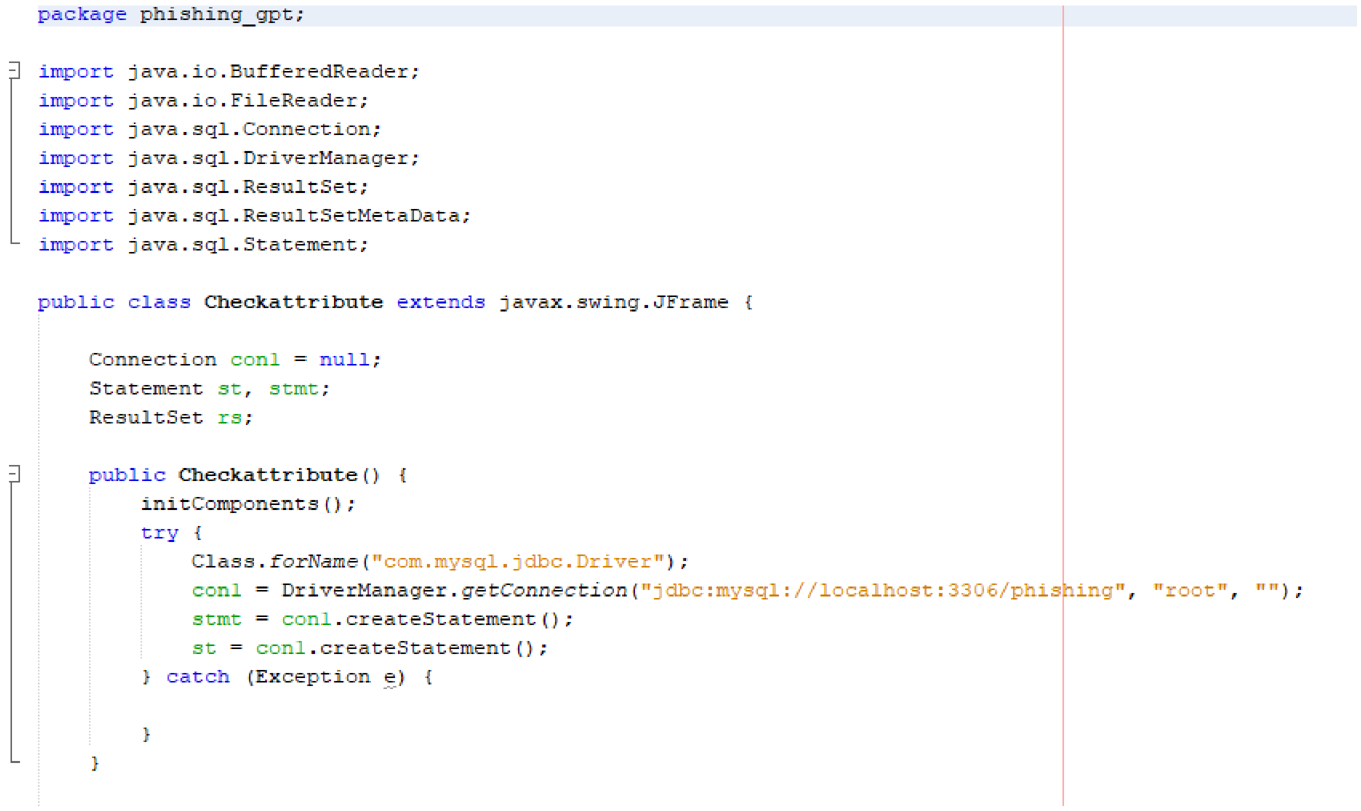
Figure 5.10: Extraction of source code of url



```java
                oc = inputLine.indexOf("<form>");
            }

            in.close();

        } catch (Exception Ex) {

        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
/         if (oc >= 1) {
/             JOptionPane.showMessageDialog(null, "<form> found \n This site may be phishing");
/         } else {
/             JOptionPane.showMessageDialog(null, "<form> tag Checking completed go further");
/         }
        new Sourcecode().setVisible(false);
        this.setVisible(false);
        new MapReduce().setVisible(true);

    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new Sourcecode().setVisible(false);
        this.setVisible(false);

    }
```

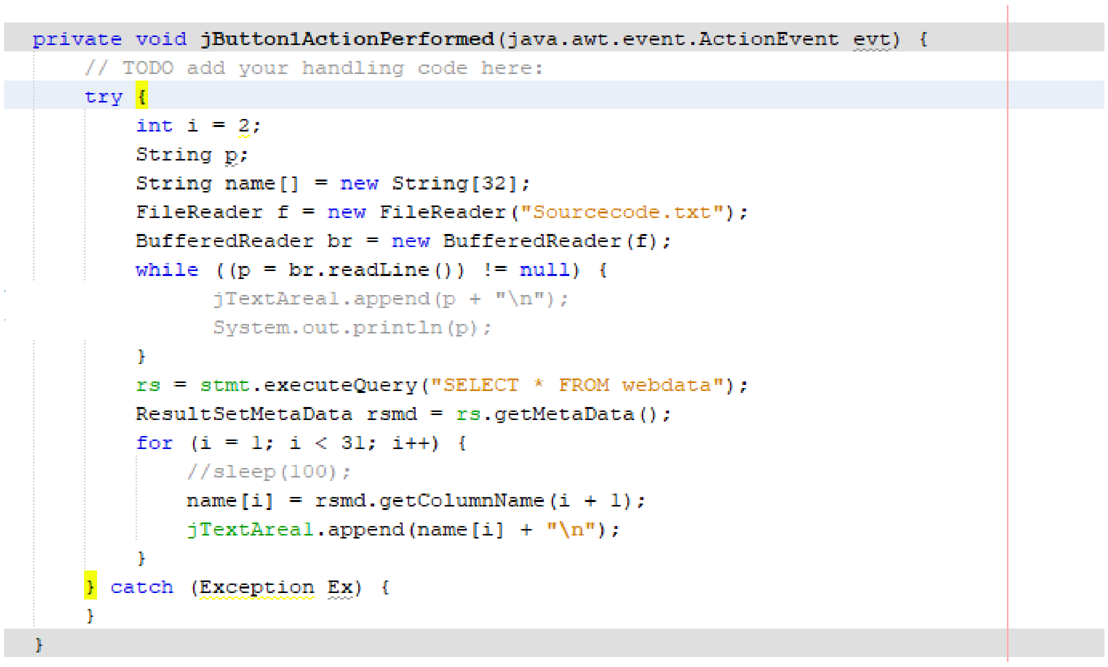
Figure 5.11: Extraction of source code of url

5.2.2 Attribute Extraction code



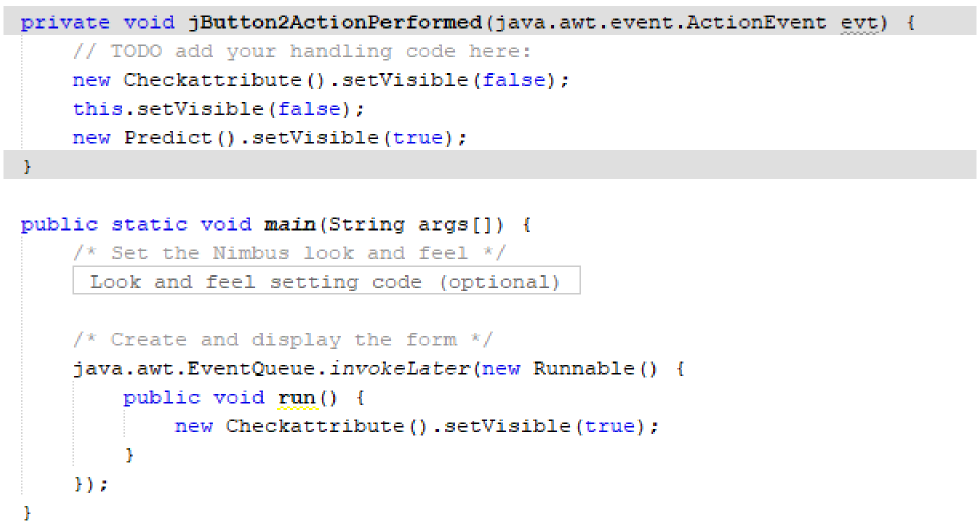
```java
package phishing_gpt;

import java.io.BufferedReader;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

public class Checkattribute extends javax.swing.JFrame {

    Connection con1 = null;
    Statement st, stmt;
    ResultSet rs;

    public Checkattribute() {
        initComponents();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con1 = DriverManager.getConnection("jdbc:mysql://localhost:3306/phishing", "root", "");
            stmt = con1.createStatement();
            st = con1.createStatement();
        } catch (Exception e) {

        }
    }
```

Figure 5.12: Attribute Extraction code



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        int i = 2;
        String p;
        String name[] = new String[32];
        FileReader f = new FileReader("Sourcecode.txt");
        BufferedReader br = new BufferedReader(f);
        while ((p = br.readLine()) != null) {
              jTextArea1.append(p + "\n");
              System.out.println(p);
        }
        rs = stmt.executeQuery("SELECT * FROM webdata");
        ResultSetMetaData rsmd = rs.getMetaData();
        for (i = 1; i < 31; i++) {
            //sleep(100);
            name[i] = rsmd.getColumnName(i + 1);
            jTextArea1.append(name[i] + "\n");
        }
    } catch (Exception Ex) {
    }
}
```

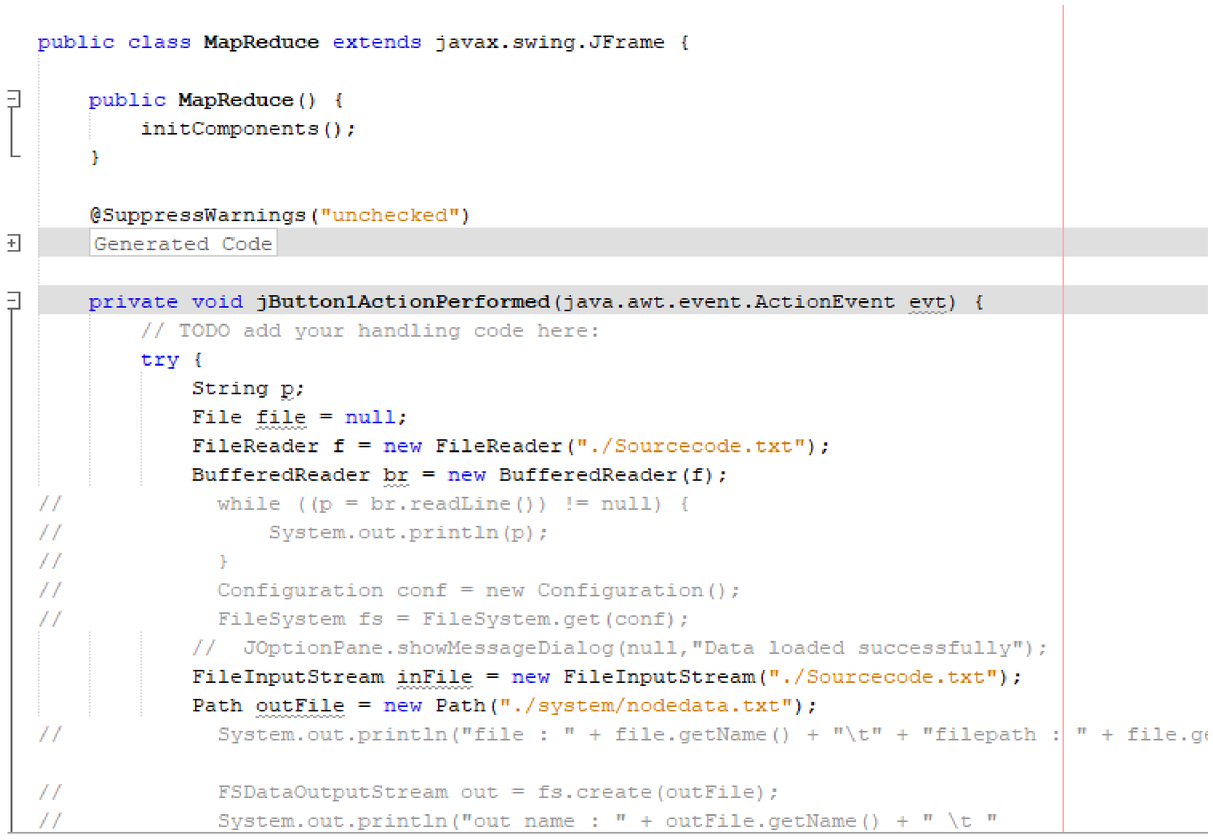
Figure 5.13: Attribute Extraction code



```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new Checkattribute().setVisible(false);
    this.setVisible(false);
    new Predict().setVisible(true);
}

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Checkattribute().setVisible(true);
        }
    });
}
```

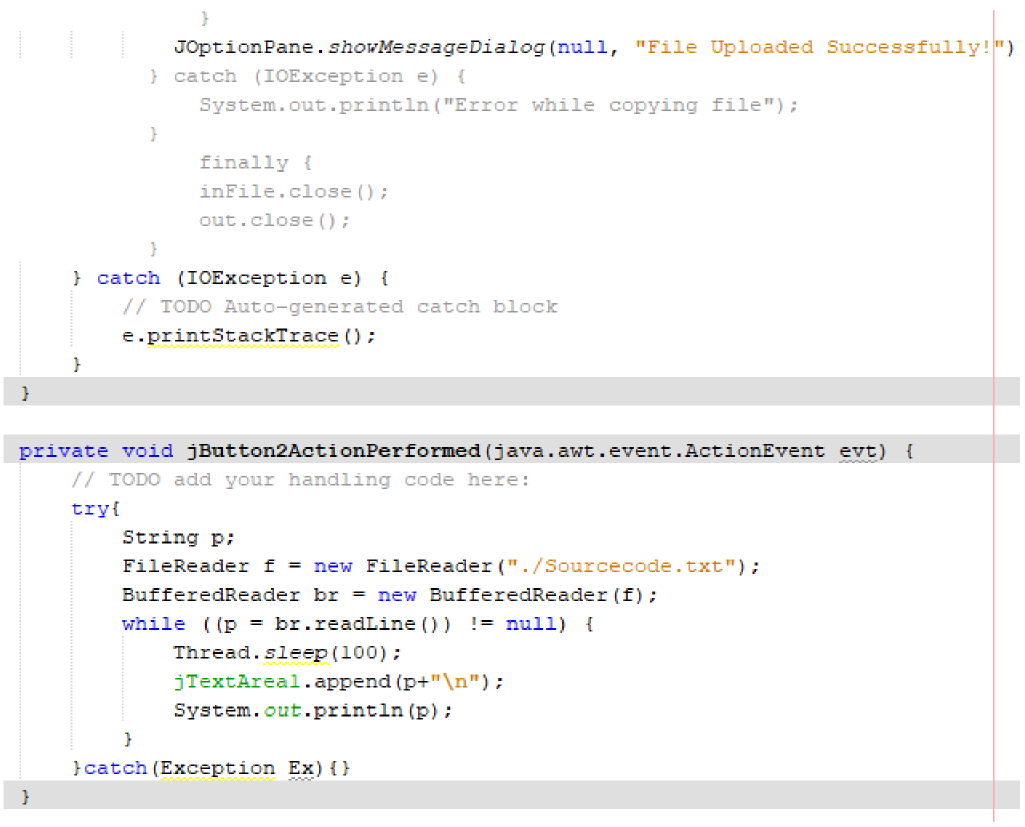
Figure 5.14: Attribute Extraction code

5.2.3 Map-Reduce code



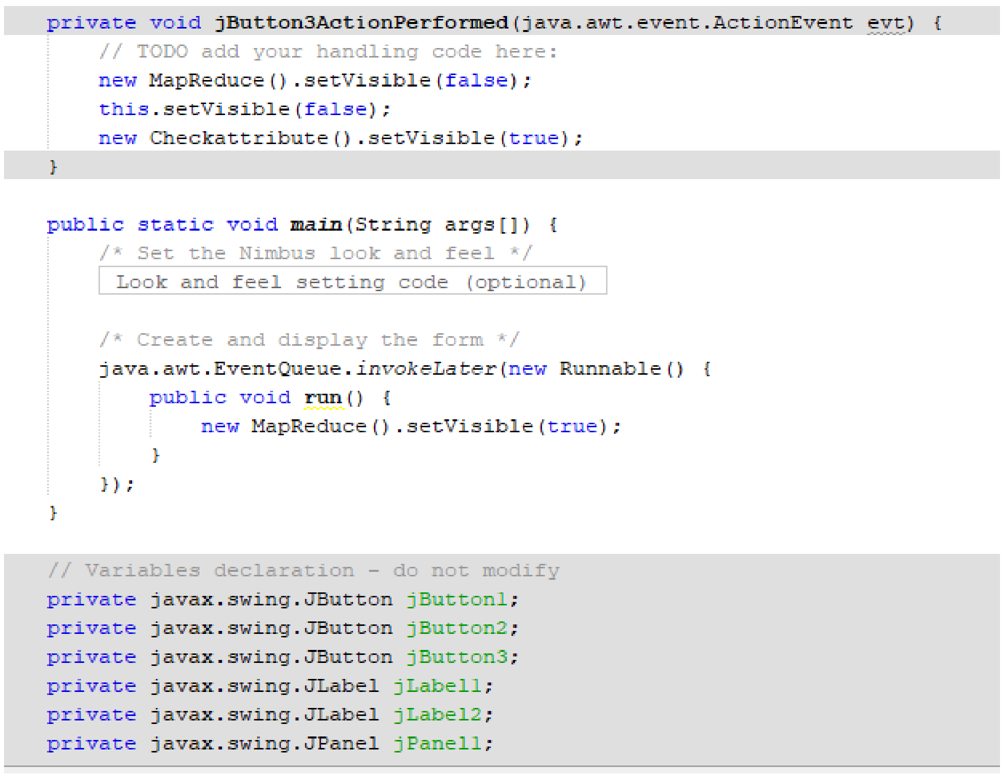
```
public class MapReduce extends javax.swing.JFrame {

    public MapReduce() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
            String p;
            File file = null;
            FileReader f = new FileReader("./Sourcecode.txt");
            BufferedReader br = new BufferedReader(f);
//              while ((p = br.readLine()) != null) {
//                  System.out.println(p);
//              }
//              Configuration conf = new Configuration();
//              FileSystem fs = FileSystem.get(conf);
            //  JOptionPane.showMessageDialog(null,"Data loaded successfully");
            FileInputStream inFile = new FileInputStream("./Sourcecode.txt");
            Path outFile = new Path("./system/nodedata.txt");
//              System.out.println("file : " + file.getName() + "\t" + "filepath : " + file.ge

//              FSDataOutputStream out = fs.create(outFile);
//              System.out.println("out name : " + outFile.getName() + " \t "
```

Figure 5.15: Applying Map-Reduce



```
                }
              JOptionPane.showMessageDialog(null, "File Uploaded Successfully!")
            } catch (IOException e) {
                System.out.println("Error while copying file");
            }
                finally {
                inFile.close();
                out.close();
            }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        String p;
        FileReader f = new FileReader("./Sourcecode.txt");
        BufferedReader br = new BufferedReader(f);
        while ((p = br.readLine()) != null) {
            Thread.sleep(100);
            jTextArea1.append(p+"\n");
            System.out.println(p);
        }
    }catch(Exception Ex){}
}
```

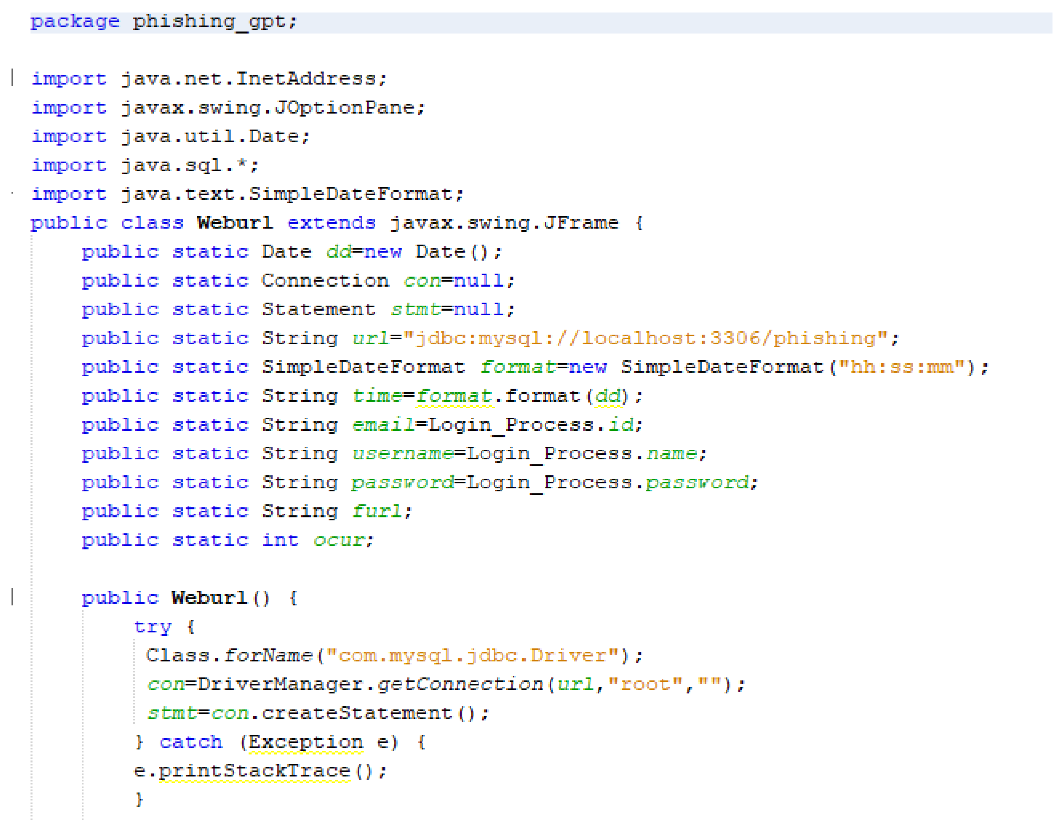
Figure 5.16: Applying Map-Reduce



```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new MapReduce().setVisible(false);
    this.setVisible(false);
    new Checkattribute().setVisible(true);
}

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new MapReduce().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JPanel jPanel1;
```

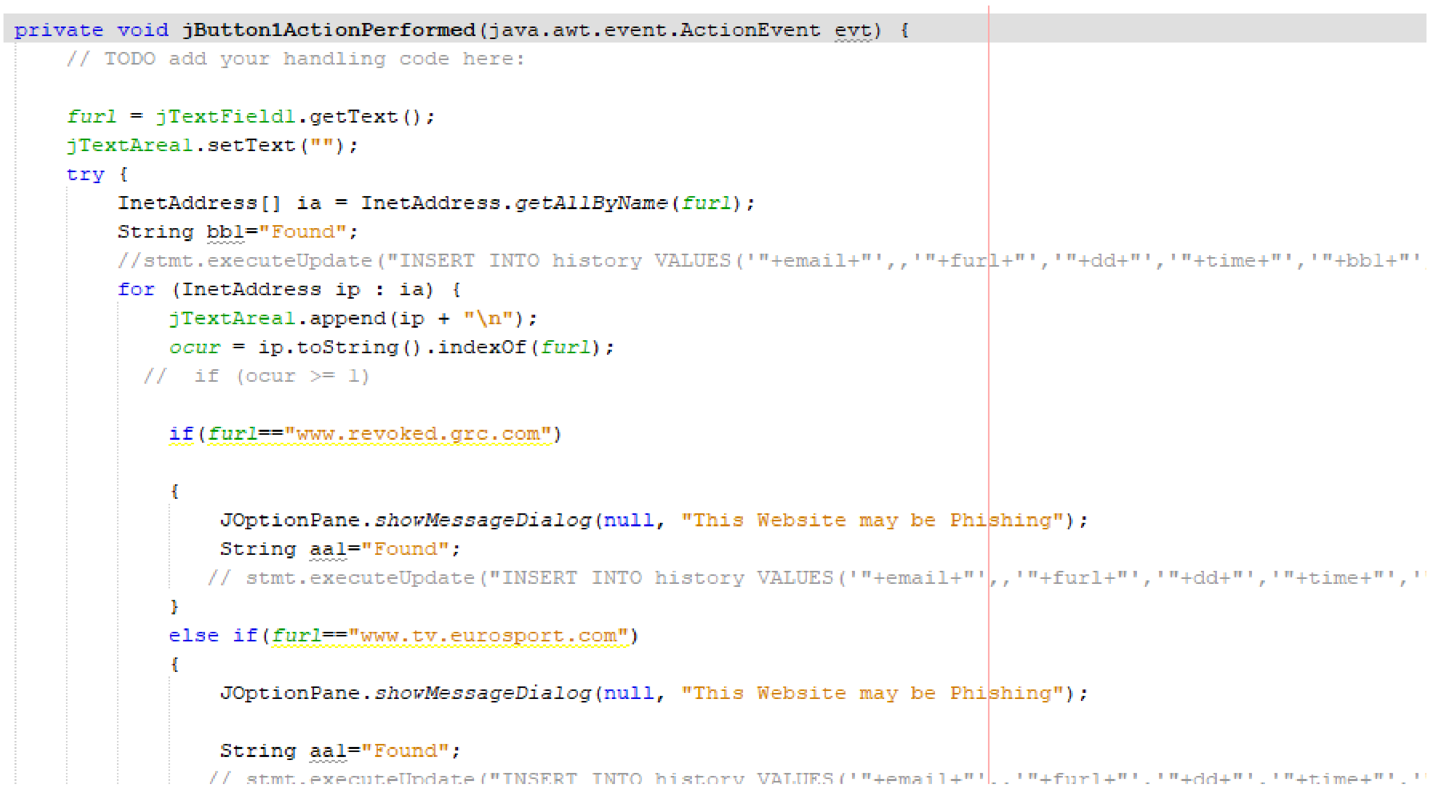
Figure 5.17: Applying Map-Reduce

5.2.4 Connection with database



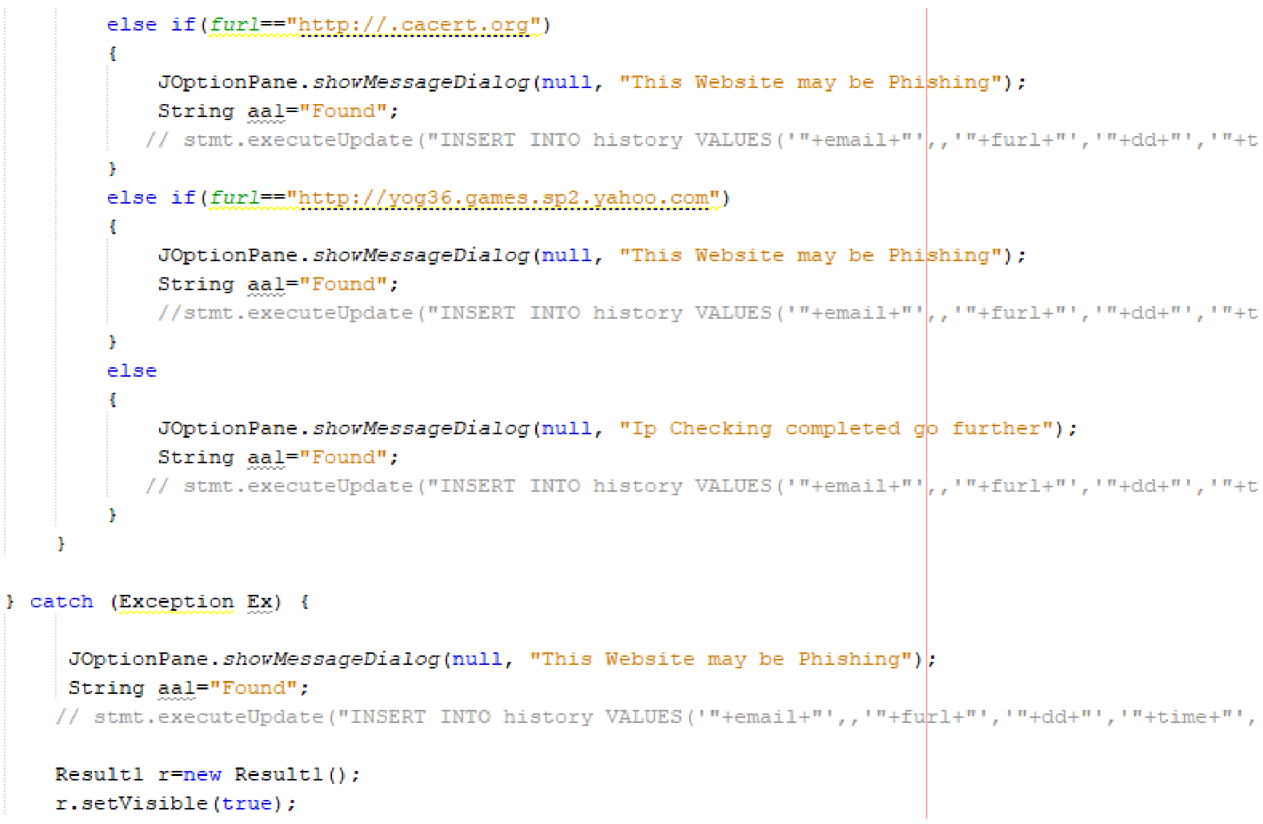
```java
package phishing_gpt;

import java.net.InetAddress;
import javax.swing.JOptionPane;
import java.util.Date;
import java.sql.*;
import java.text.SimpleDateFormat;
public class Weburl extends javax.swing.JFrame {
    public static Date dd=new Date();
    public static Connection con=null;
    public static Statement stmt=null;
    public static String url="jdbc:mysql://localhost:3306/phishing";
    public static SimpleDateFormat format=new SimpleDateFormat("hh:ss:mm");
    public static String time=format.format(dd);
    public static String email=Login_Process.id;
    public static String username=Login_Process.name;
    public static String password=Login_Process.password;
    public static String furl;
    public static int ocur;

    public Weburl() {
        try {
         Class.forName("com.mysql.jdbc.Driver");
         con=DriverManager.getConnection(url,"root","");
         stmt=con.createStatement();
        } catch (Exception e) {
        e.printStackTrace();
        }
```

Figure 5.18: Connecting database using WAMP



```java
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    furl = jTextField1.getText();
    jTextArea1.setText("");
    try {
        InetAddress[] ia = InetAddress.getAllByName(furl);
        String bbl="Found";
        //stmt.executeUpdate("INSERT INTO history VALUES('"+email+"',,'"+furl+"','"+dd+"','"+time+"','"+bbl+"'
        for (InetAddress ip : ia) {
            jTextArea1.append(ip + "\n");
            ocur = ip.toString().indexOf(furl);
          //  if (ocur >= 1)

            if(furl=="www.revoked.grc.com")

            {
                JOptionPane.showMessageDialog(null, "This Website may be Phishing");
                String aal="Found";
              // stmt.executeUpdate("INSERT INTO history VALUES('"+email+"',,'"+furl+"','"+dd+"','"+time+"','
            }
            else if(furl=="www.tv.eurosport.com")
            {
                JOptionPane.showMessageDialog(null, "This Website may be Phishing");

                String aal="Found";
              // stmt.executeUpdate("INSERT INTO history VALUES('"+email+"',,'"+furl+"','"+dd+"','"+time+"','
```

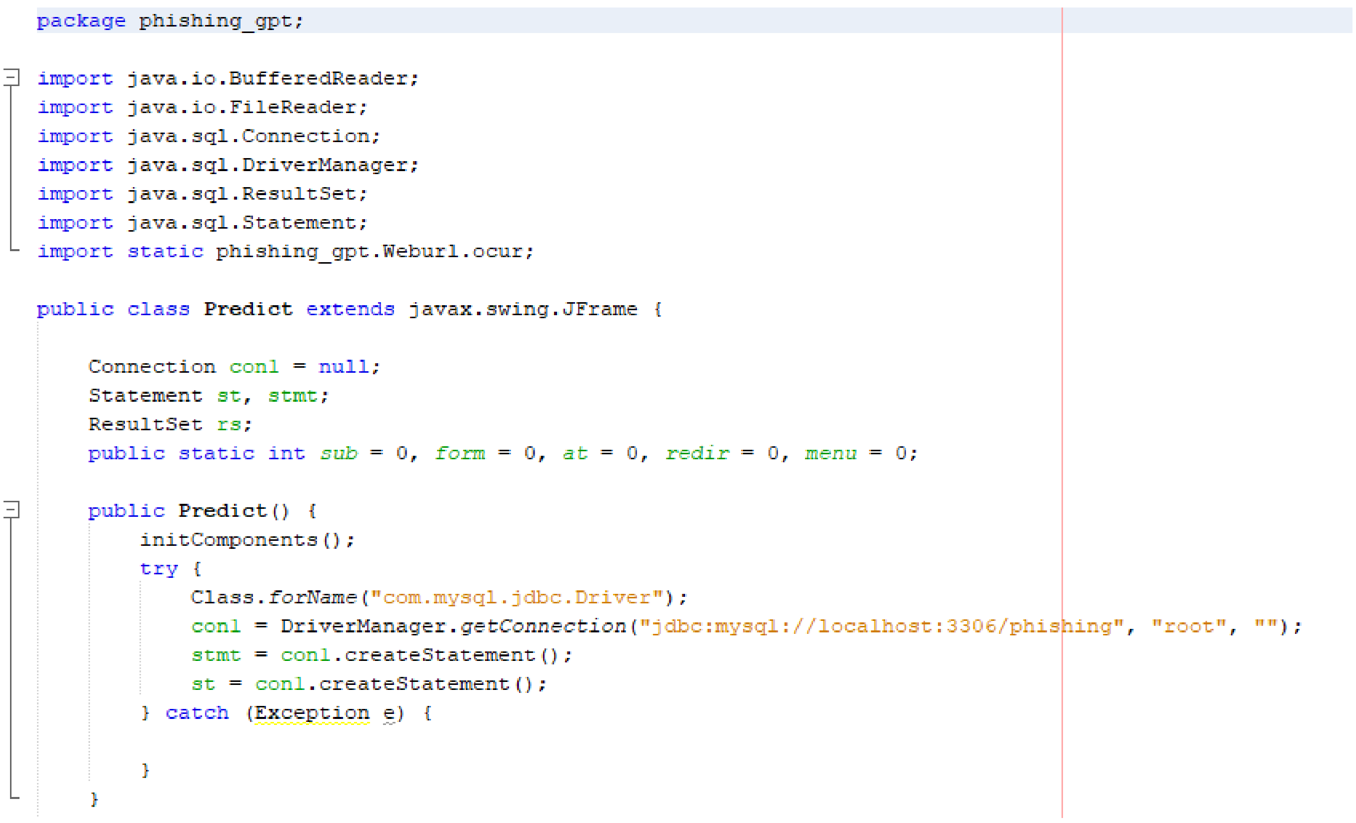
Figure 5.19: Connecting database using WAMP



```
            else if(furl=="http://.cacert.org")
            {
                JOptionPane.showMessageDialog(null, "This Website may be Phishing");
                String aal="Found";
               // stmt.executeUpdate("INSERT INTO history VALUES('"+email+"','"+furl+"','"+dd+"','"+t
            }
            else if(furl=="http://yog36.games.sp2.yahoo.com")
            {
                JOptionPane.showMessageDialog(null, "This Website may be Phishing");
                String aal="Found";
                //stmt.executeUpdate("INSERT INTO history VALUES('"+email+"','"+furl+"','"+dd+"','"+t
            }
            else
            {
                JOptionPane.showMessageDialog(null, "Ip Checking completed go further");
                String aal="Found";
               // stmt.executeUpdate("INSERT INTO history VALUES('"+email+"','"+furl+"','"+dd+"','"+t
            }
        }

} catch (Exception Ex) {

      JOptionPane.showMessageDialog(null, "This Website may be Phishing");
      String aal="Found";
     // stmt.executeUpdate("INSERT INTO history VALUES('"+email+"','"+furl+"','"+dd+"','"+time+"',

     Result1 r=new Result1();
     r.setVisible(true);
```

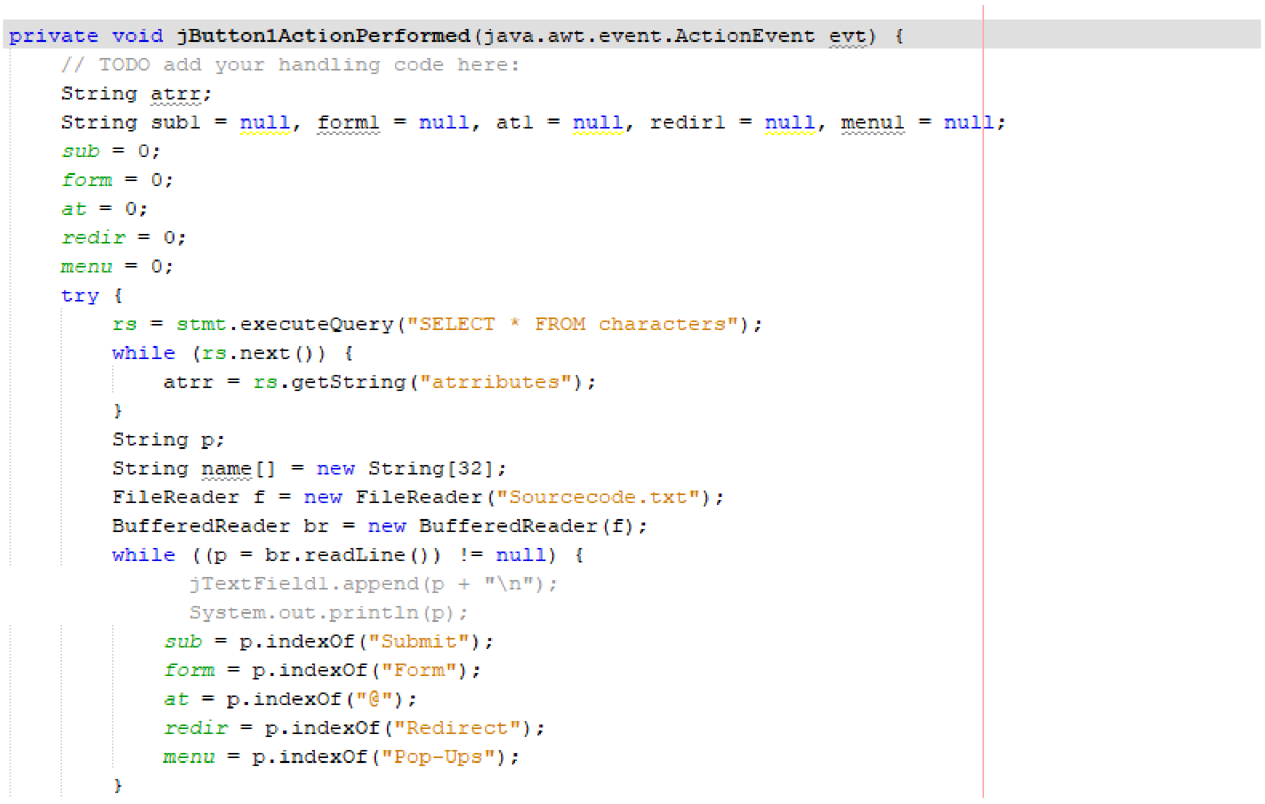
Figure 5.20: Connecting database using WAMP

5.2.5 Prediction Code



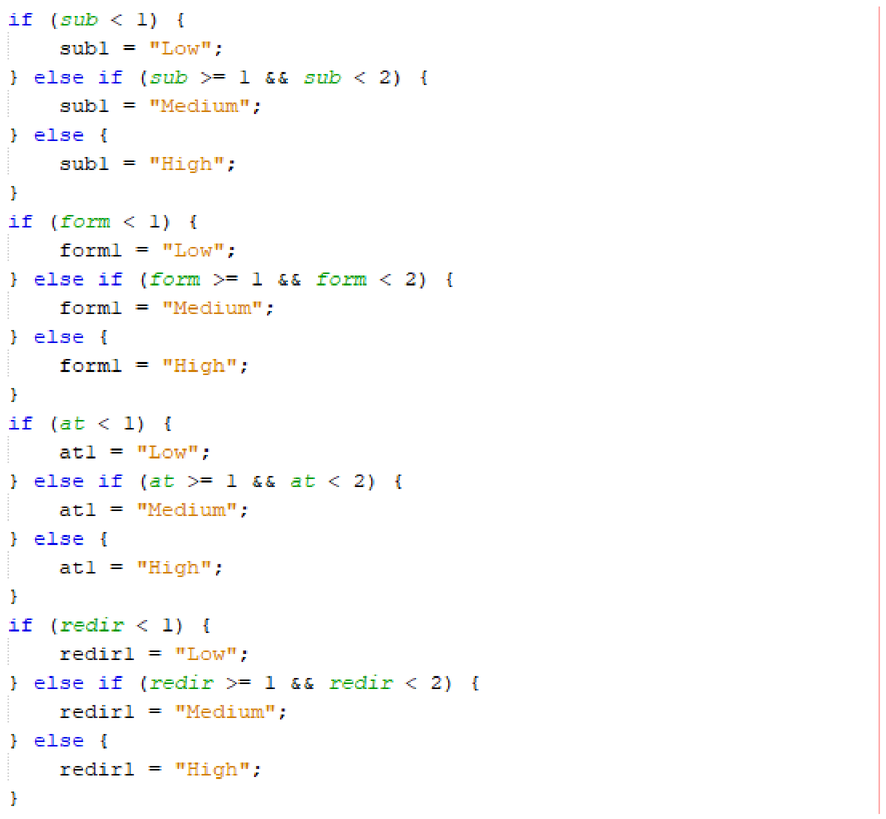
```
package phishing_gpt;

import java.io.BufferedReader;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import static phishing_gpt.Weburl.ocur;

public class Predict extends javax.swing.JFrame {

    Connection con1 = null;
    Statement st, stmt;
    ResultSet rs;
    public static int sub = 0, form = 0, at = 0, redir = 0, menu = 0;

    public Predict() {
        initComponents();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con1 = DriverManager.getConnection("jdbc:mysql://localhost:3306/phishing", "root", "");
            stmt = con1.createStatement();
            st = con1.createStatement();
        } catch (Exception e) {

        }
    }
```

Figure 5.21: Prediction Using Machine Learning



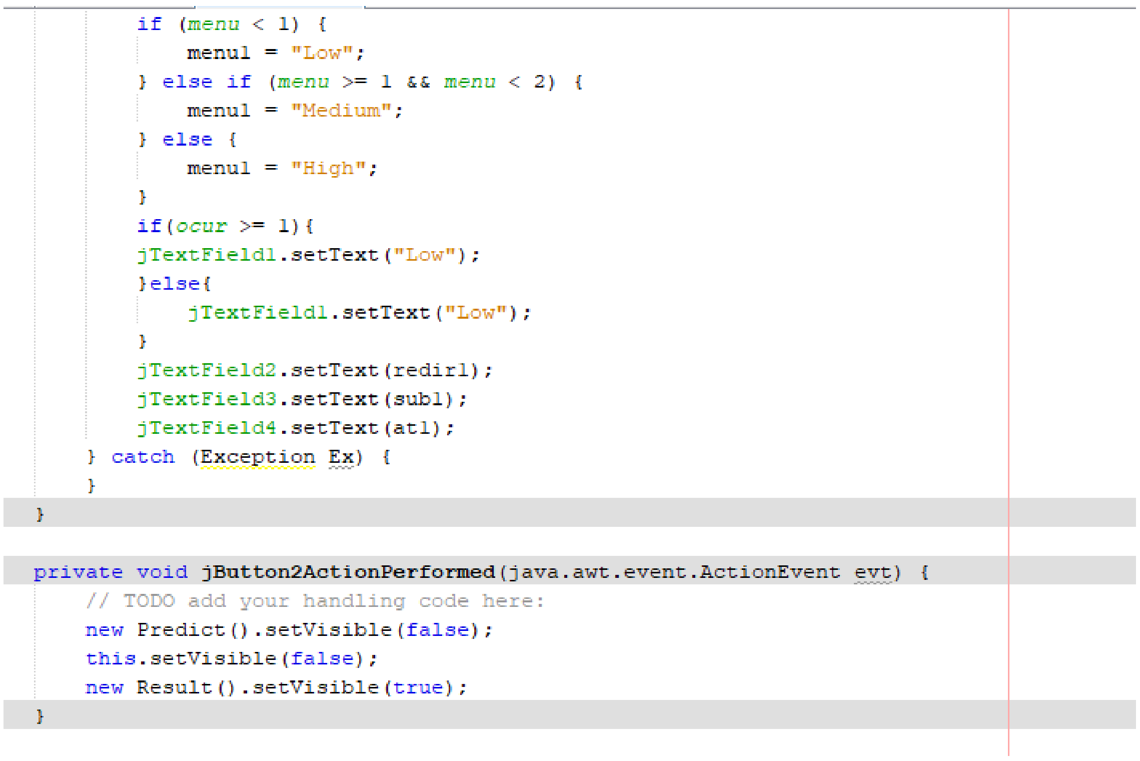
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String atrr;
    String sub1 = null, form1 = null, at1 = null, redir1 = null, menu1 = null;
    sub = 0;
    form = 0;
    at = 0;
    redir = 0;
    menu = 0;
    try {
        rs = stmt.executeQuery("SELECT * FROM characters");
        while (rs.next()) {
            atrr = rs.getString("atrributes");
        }
        String p;
        String name[] = new String[32];
        FileReader f = new FileReader("Sourcecode.txt");
        BufferedReader br = new BufferedReader(f);
        while ((p = br.readLine()) != null) {
              jTextField1.append(p + "\n");
              System.out.println(p);
            sub = p.indexOf("Submit");
            form = p.indexOf("Form");
            at = p.indexOf("@");
            redir = p.indexOf("Redirect");
            menu = p.indexOf("Pop-Ups");
        }
```

Figure 5.22: Prediction Using Machine Learning



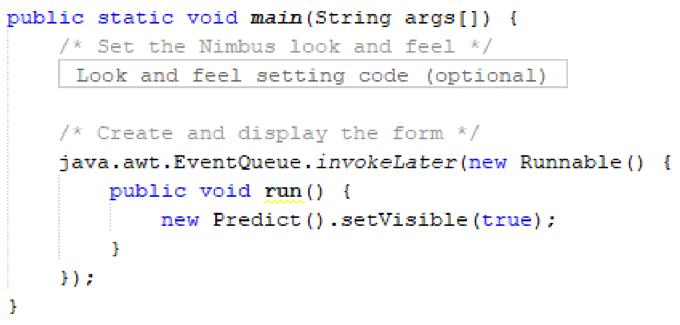
```
if (sub < 1) {
    sub1 = "Low";
} else if (sub >= 1 && sub < 2) {
    sub1 = "Medium";
} else {
    sub1 = "High";
}
if (form < 1) {
    form1 = "Low";
} else if (form >= 1 && form < 2) {
    form1 = "Medium";
} else {
    form1 = "High";
}
if (at < 1) {
    at1 = "Low";
} else if (at >= 1 && at < 2) {
    at1 = "Medium";
} else {
    at1 = "High";
}
if (redir < 1) {
    redir1 = "Low";
} else if (redir >= 1 && redir < 2) {
    redir1 = "Medium";
} else {
    redir1 = "High";
}
```

Figure 5.23: Prediction Using Machine Learning



```java
                if (menu < 1) {
                    menul = "Low";
                } else if (menu >= 1 && menu < 2) {
                    menul = "Medium";
                } else {
                    menul = "High";
                }
                if(ocur >= 1){
                jTextField1.setText("Low");
                }else{
                    jTextField1.setText("Low");
                }
                jTextField2.setText(redirl);
                jTextField3.setText(subl);
                jTextField4.setText(atl);
            } catch (Exception Ex) {
            }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new Predict().setVisible(false);
        this.setVisible(false);
        new Result().setVisible(true);
    }
```

Figure 5.24: Prediction Using Machine Learning



```java
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Predict().setVisible(true);
        }
    });
}
```

Figure 5.25: Prediction Using Machine Learning

5.3 Testing

5.3.1 Unit Testing

Unit Testing: The goal of unit testing to separate each part of the program and test that the individual parts are working correctly and as intended. Test Objectives: Properly the system is being able to accept the url and is able to perform map and reduce function.

Test Condition	**Input Specification**	**Output specification**	**Success/Fail**
URLs accepted by system	User Enters URL	IP Address	Success

Table 5.1: Unit Testing

5.3.2 Integration Testing

Integration Testing: It is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
Test Objectives: The urls which are been analysed using map reduce function are now predicted whether its phished or not phished.

Test Condition	**Input Specification**	**Output specification**	**Success/Fail**
Predicts the nature of URL	User Enters URL	System Predicts	Success

Table 5.2: Integration Testing

5.3.3 Blackbox Testing

Black box Testing: In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behavior of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.
Test Objectives: Here the user need to do trial and error by put in all types of urls into the system to check if the system is fully functional. When a phished Url is been fed into the system if it is not successfully showing that it is a phishing website then improvements need to be made, also when genuine url is being added and its not taking to the genuine website then also required changes need to be done.

Test Condition	**Input Specification**	**Output specification**	**Success/Fail**
The system is able to predict or not	User Enters URL	phishing website or genuine website	Success

Table 5.3: Blackbox Testing

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

Main goal of the system is to achieve speed up in existing anti-phishing system by some means. Using Hadoop-MapReduce in integration with anti-phishing technique we have achieved considerable time speedup. Even if the phishing webpage is not showing phishing characteristics very clearly at first layer it might show characteristics in the next layer so that no phishing webpage will pass through our system. This is the advantage of having layered architecture of attributes. Hadoop-MapReduce will increase the response time of the system considerably. This system is very effective in securing network from phishing attach even at its best.

6.2 Future Scope

There is a lot of scope for improvement of this system. One can improve the performance of system by converting this as a cloud service. As per type of organization we are protecting from phishing attack change the attributes to be considered for making effective decisions about the phishiness of the system.

References

[1] E. P. Pujara and M. Chaudhari, "Phishing website detection using machine learning : A review," 02 2019.

[2] U. K. Sah and N. S. Parmar, "An approach for malicious spam detection in email with comparison of different classifiers," 2017.

[3] V. Khamkar, P. Ingale, and D. D. Walunj, "Anti-phishing using big data," 2018.

[4] M. Gavahane, D. Sequeira, A. Pandey, and A. Shetty, "Anti-phishing using hadoop-framework," in *2015 International Conference on Technologies for Sustainable Development (ICTSD)*, pp. 1–4, Feb 2015.

[5] P. Patil, R. Rane, and M. Bhalekar, "Detecting spam and phishing mails using svm and obfuscation url detection algorithm," in *2017 International Conference on Inventive Systems and Control (ICISC)*, pp. 1–4, Jan 2017.

[6] S. Thakur, E. Meenakshi, and A. Priya, "Detection of malicious urls in big data using ripper algorithm," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pp. 1296–1301, May 2017.

[7] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani, "Systematization of knowledge (sok): A systematic review of software-based web phishing detection," *IEEE Communications Surveys Tutorials*, vol. 19, pp. 2797–2819, Fourthquarter 2017.

[8] S. Parekh, D. Parikh, S. Kotak, and S. Sankhe, "A new method for detection of phishing websites: Url detection," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 949–952, April 2018.

[9] R. Aravindhan, R. Shanmugalakshmi, K. Ramya, and Selvan C., "Certain investigation on web application security: Phishing detection and phishing target discovery," in *2016 3rd International Conference on*

Advanced Computing and Communication Systems (ICACCS), vol. 01, pp. 1–10, Jan 2016.

[10] H. Sharma, E. Meenakshi, and S. K. Bhatia, “A comparative analysis and awareness survey of phishing detection tools,” in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pp. 1437–1442, May 2017.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10]

Acknowledgement

Success of a project like this involving high technical expertise, patience and massive support of guides, is possible when team members work together. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to show our appreciation to **Mr. Mritunjay Ojha** for their tremendous support and help, without them this project would have reached nowhere. We would also like to thank our project coordinator **Mrs. Rakhi Kalantri** for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Lata Ragha** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, **Fr. C. Rodrigues Institute of Technology, Vashi**, for giving us the opportunity and the environment to learn and grow.

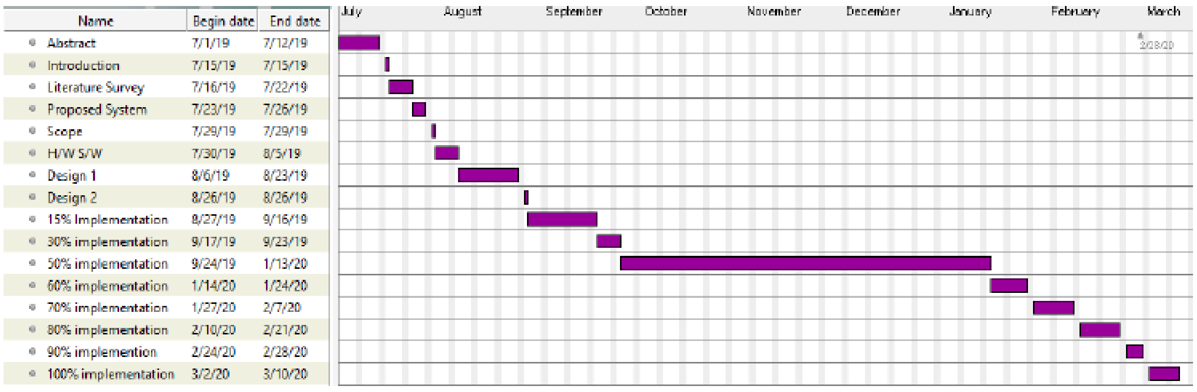
Project Group Members:

1. A. Annie Grace, 101601

2. Tanya Serah Jacob, 101656

3. Swapnil Monteiro, 101670

Appendix A : Timeline Chart



Name	Begin date	End date
Abstract	7/1/19	7/12/19
Introduction	7/15/19	7/15/19
Literature Survey	7/16/19	7/22/19
Proposed System	7/23/19	7/26/19
Scope	7/29/19	7/29/19
H/W S/W	7/30/19	8/5/19
Design 1	8/6/19	8/23/19
Design 2	8/26/19	8/26/19
15% Implementation	8/27/19	9/16/19
30% implementation	9/17/19	9/23/19
50% implementation	9/24/19	1/13/20
60% implementation	1/14/20	1/24/20
70% implementation	1/27/20	2/7/20
80% implementation	2/10/20	2/21/20
90% implemention	2/24/20	2/28/20
100% implementation	3/2/20	3/10/20

Figure 6.1: Timeline chart

Appendix B : Publication Details

6.3 Publication Details

Annie Grace Anthony, Tanya Serah Jacob, Swapnil Monteiro, International Journal of Research and Technology (IJRT), ISSN:2321-7510 (Print) — 2321-7529 (Online), Volume 8, Issue 2, Page No:186-189