

TITLE: DETECTION OF E BANKING PHISHING WEBSITES

TEAM:

SARVEPALLI MOHITH -----18BCE0013

PATHAPATI SAI SARAVAN-----18BCE0088

ABSTRACT:

E-banking phishing websites are forged websites that are created by malicious people to mimic real e-banking websites. Most of these kinds of web pages have high visual similarities to scam their victims. Some of these web pages look exactly like the real ones. Unwary Internet users may be easily deceived by this kind of scam. Victims of e-banking phishing websites may expose their bank account, password, credit card number, or other important information to the phishing web page owners. The impact is the breach of information security through the compromise of confidential data, and the victims may finally suffer losses of money or other kinds. Phishing is a relatively new Internet crime in comparison with other forms, e.g., virus and hacking. E-banking phishing website is a very complex issue to understand and to analyze, since it is joining technical and social problem with each other for which there is no known single silver bullet to entirely solve it. The motivation behind this study is to create a resilient and effective method that uses associative classification algorithms and tools to detect e-banking phishing websites in an automated manner. The concept is an end-host based anti-phishing algorithm, called the Link Guard, by utilizing the generic characteristics of the hyperlinks in phishing attacks. The link Guard algorithm is the concept for finding the phishing emails sent by the phisher to grasp the information of the end user. Link Guard is based on the careful analysis of the characteristics of phishing hyperlinks. Each end user is implemented with Link Guard algorithm. After doing so the end user recognizes the phishing emails and can avoid responding to such mails. Since Link Guard is characteristics based it can detect and prevent not only known phishing attacks but also unknown ones. The project uses the ASP .NET as Front-End and SQL as Back-End. [Security and Privacy]: Intrusion/anomaly detection and malware mitigation - Social engineering attacks ,phishing ; Security Services-authentication, access control.

KEYWORDS:

E-BANKING : Electronic banking is a form of banking in which funds are transferred through an exchange of electronic signals rather than through an exchange of cash, checks, or other types of paper documents. Transfers of funds occur between financial institutions such as banks and credit unions.

PHISHING: Phishing is a type of social engineering attack often used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.

GRAPHICAL USER INTERFACE : It is a user interface that includes graphical elements, such as windows, icons and buttons. The term was created in the 1970s to distinguish graphical interfaces from text-based ones, such as command line interfaces.

.NET FRAMEWORK: The .NET framework is a software development framework from Microsoft. It provides a controlled programming environment where software can be developed, installed and executed on Windows-based operating systems. ... Security: Programs developed in .NET are based on a common security model.

DNS :DNS. (Domain Name System) The Internet's system for converting alphabetic names into numeric IP addresses. For example, when a Web address (URL) is typed into a browser, DNS servers return the IP address of the Web server associated with that name.

URL: URL stands for Uniform Resource Locator, and is used to specify addresses on the World Wide Web. A URL is the fundamental network identification for any resource connected to the web (e.g., hypertext pages, images, and sound files). ... The domain name is the computer on which the resource is located

BLACK LISTING :Blacklisting is the action of a group or authority, compiling a blacklist (or black list) of people, countries or other entities to be avoided or distrusted as not being acceptable to those making the list. A blacklist can list people to be discriminated against, refused employment, or censored.

Introduction

Phishing is a word produced from 'fishing', it refers to the act that the attacker allure users to visit a fake Web site by sending them fake e-mails (or instant messages), and stealthily get victim's personal information such as user name, password, and national security ID, etc. This information then can be used for future target advertisements or even identity theft attacks (e.g., transfer money from victims' bank account). The frequently used attack method is to send e-mails to potential victims, which seemed to be sent by banks, online organizations, or ISPs. In these e-mails, they will make up some causes, e.g. the password of your credit card had been miss-entered for many times, or they are providing upgrading services, to allure you to visit their Web site to conform or modify your account number and password through the hyperlink provided in the e-mail. If you input the account number and password, the attackers then successfully collect the information at the server side, and is able to perform their next step actions with that information (e.g., withdraw money out from your account). Phishing itself is not a new concept, but it's increasingly used by phishers to steal user information and perform business crime in recent years. Within one to two years, the number of phishing attacks increased dramatically. The analysis done in the base paper identifies that the phishing hyperlinks share one or more characteristics as listed below:

1. The visual link and the actual link are not the same.
2. The attackers often use dotted decimal IP address instead of DNS name.

3. Special tricks are used to encode the hyperlinks maliciously.
4. The attackers often use fake DNS names that are similar (but not identical) with the target Web site.

Literature Review

Phishing website is a recent problem, nevertheless due to its huge impact on the financial and on-line retailing sectors and since preventing such attacks is an important step towards defending against e-banking phishing website attacks, there are several promising defending approaches to this problem reported earlier. In this section, we briefly survey existing anti-phishing solutions and list of the related works.

One approach is to stop phishing at the e-mail level [1] (Adida, Hohenberge, Rivest, 2005), since most current phishing attacks use broadcast e-mail (spam) to lure victims to a phishing website [2] (Wu, Miller, Garfinkel, 2006a). Another approach is to use security toolbars. The phishing filter in [1] IE7 (Sharif, 2006) is a toolbar approach with more features such as blocking the users activity with a detected phishing site. A third approach is to visually differentiate the phishing sites from the spoofed legitimate sites. Dynamic Security Skins [1] (Dhamija and Tygar, 2005) proposes to use a randomly generated visual hash to customize the browser window or web form elements to indicate the successfully authenticated sites. A fourth approach is two-factor authentication, which ensures that the user not only knows a secret but also presents a security token (FDIC, 2004). However, this approach is a server-side solution. Phishing can still happen at sites that do not support two-factor authentication. Sensitive information that is not related to a specific site, e.g., credit card information and SSN (Social Security Number), cannot be protected by this approach either [2] (Wu, Miller, and Little, 2006b). Some researchers have shown that security tool bars do not effectively prevent phishing attacks. [3] Bridges and Vaughn (2001), [1] Dhamija and Tygar (2005) proposed a scheme that utilises a cryptographic identity-verification method that lets remote web servers prove their identities. However, this proposal requires changes to the entire web infrastructure (both servers and clients), so it can succeed only if the entire industry supports it. In [4] Liu, Deng, Huang, and Fu (2006), the authors proposed a tool to model and describe phishing by visualizing and quantifying a given sites threat, but this method still would not provide an anti-phishing solution. Another approach is to employ certification, e.g., Microsoft spam privacy (Microsoft, 2004; Microsoft Corp., 2005; Olsen, 2004; Perez, 2003; Whole Security Web Caller, 2005). A recent and particularly promising solution was proposed in [1] Herzberg and Gbara (2004), which combines the technique of standard certificates with a visual indication of correct certification; a site dependent logo indicating that the certificate was valid would be displayed in a trusted credentials area of the browser. A variant of web credential is to use a database or list published by a trusted party, where known phishing websites are blacklisted. For example, Netcraft anti-phishing toolbar (Netcraft, 2004) prevents phishing attacks by utilising a centralized blacklist of current phishing URLs. Other examples include websense, McAfee's anti-phishing filter, Netcraft anti-phishing system, Cloudmark SafetyBar, and Microsoft Phishing Filter [5] (Pan and Ding, 2006). The weaknesses of this approach are its poor scalability and its timeliness. Note that phishing sites are cheap and easy to build and their average lifetime is only a few days. APWG provides a solution directory at Anti-Phishing Working Group (2007) which contains most of the major anti-phishing companies in the

world. However, an automatic anti-phishing method is seldom reported. The typical technologies of anti-phishing from the user interface aspect are done by [1] Dhamija and Tygar (2005) and [2] Wu et al., 2006b. They proposed methods that need web page creators to follow certain rules to create web pages, either by adding dynamic skin to web pages or adding sensitive information location attributes to HTML code. However, it is difficult to convince all web page creators to follow the rules [1](Fu et al., 2006). In [1]Fu et al. (2006), [2] Liu, Huang, [2]Liu, Zhang, and Deng, 2005; [2]Liu et al., 2006), the DOM-based (Wood, 2005) visual similarity of web pages is oriented, and the concept of visual approach to phishing detection was first introduced. Through this approach, a phishing web page can be detected and reported in an automatic way rather than involving too many human efforts. Their method first decomposes the web pages (in HTML) into salient (visually distinguishable) block regions. The visual similarity between two web pages is then evaluated in three metrics: block level similarity, layout similarity, and overall style similarity, which are based on the matching of the salient block regions

EXISTING WORK

Evolving with the anti-Phishing techniques, various phishing techniques and more complicated and hard-to detect methods are used by phishers. The most straightforward way for a phisher to defraud people is to make the phishing Web pages similar to their targets. Actually, there are many characteristics and factors that can distinguish the original legitimate website from the forged e-banking phishing website like Spelling errors, Long URL address and Abnormal DNS record. The full list is shown in table I which is used later on our analysis and methodology study.

In this model we are using the detection process using URL using real time algorithms. This can be considered as basic functionality detection but it has a huge success rate because most of the hackers try to cheat using morphed look alike websites.

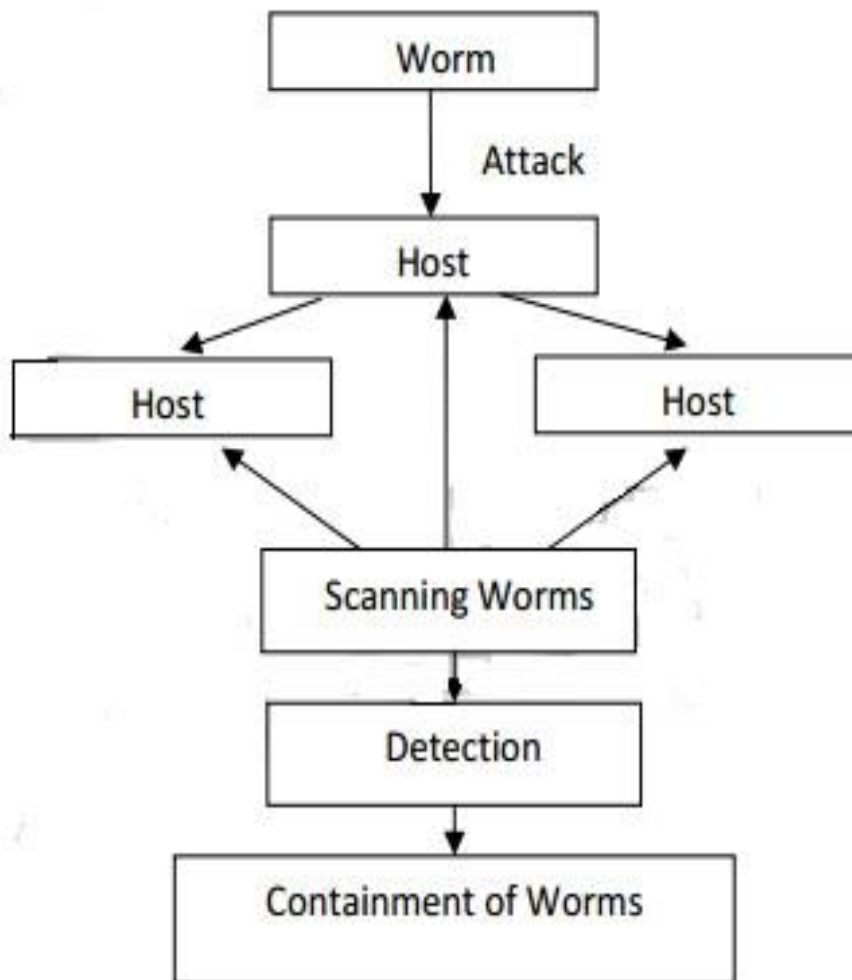
We use LINK-GUARD ALGORITHM , it is an end-host based .

The Link Guard algorithm works as follows. In its main routine LinkGuard, it first extracts the DNS names from the actual and the visual links . It then compares the actual and visual DNS names, if these names are not the same, then it is phishing of category 1 .If dotted decimal IP address is directly used in actual DNS, it is then a possible phishing attack of category 2 .If the actual link or the visual link is encoded.(Categories 3 and 4), we first decode the links, then recursively call Link Guard to return a result . When there is no destination information (DNS name or dotted IP address) in the visual link (category 5), Link Guard calls Analyses to analyse the actual DNS . Link Guard therefore handles all the 5 categories of phishing attacks.

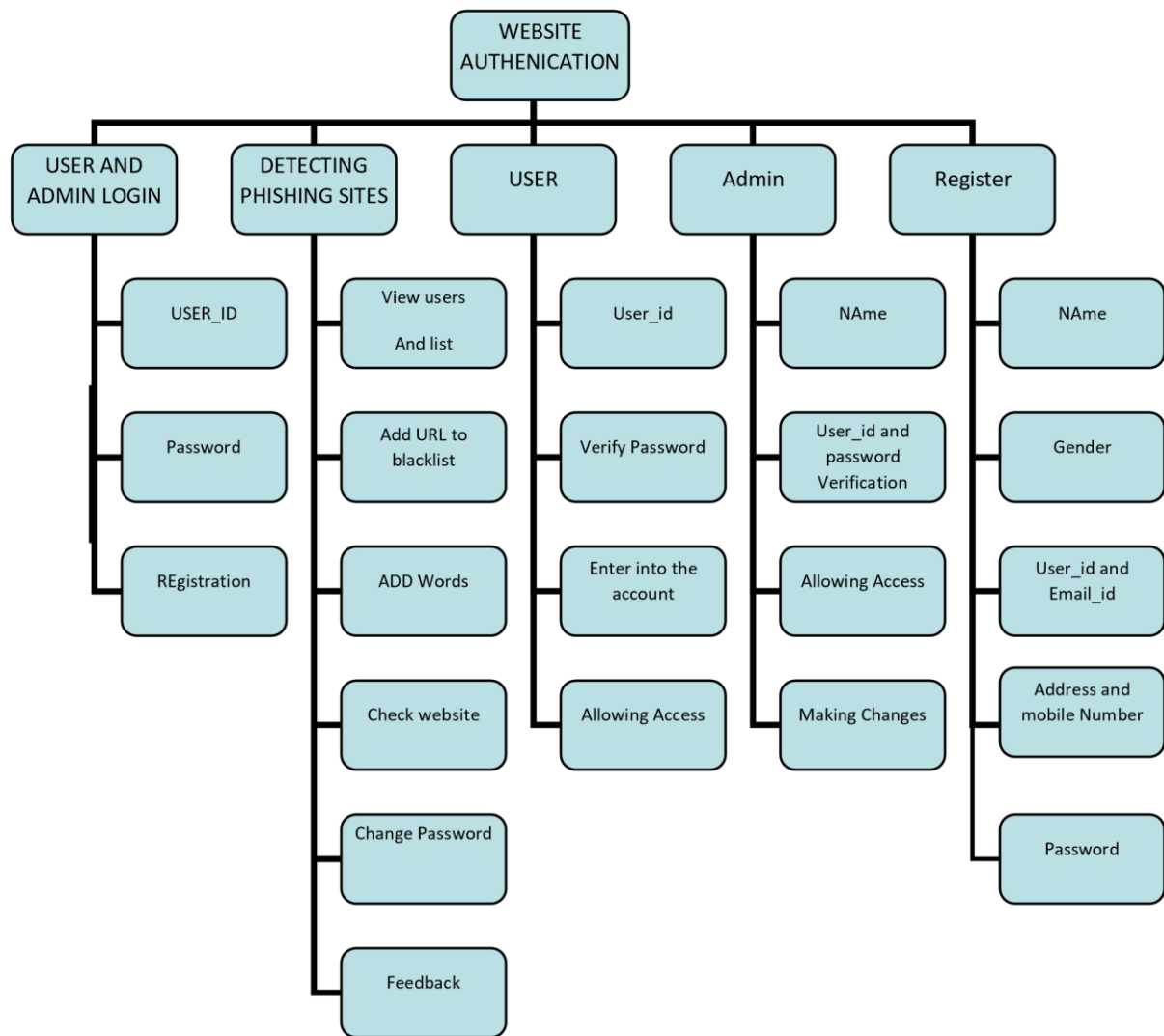
In Analyse DNS, if the actual DNS name is contained in the blacklist, then we are sure that it is a phishing attack . Similarly, if the actual DNS is contained in the whitelist, it is therefore not a phishing attack . If the actual DNS is not contained in either whitelist or blacklist, Pattern Matching is then invoked.

Proposed SYSTEM:

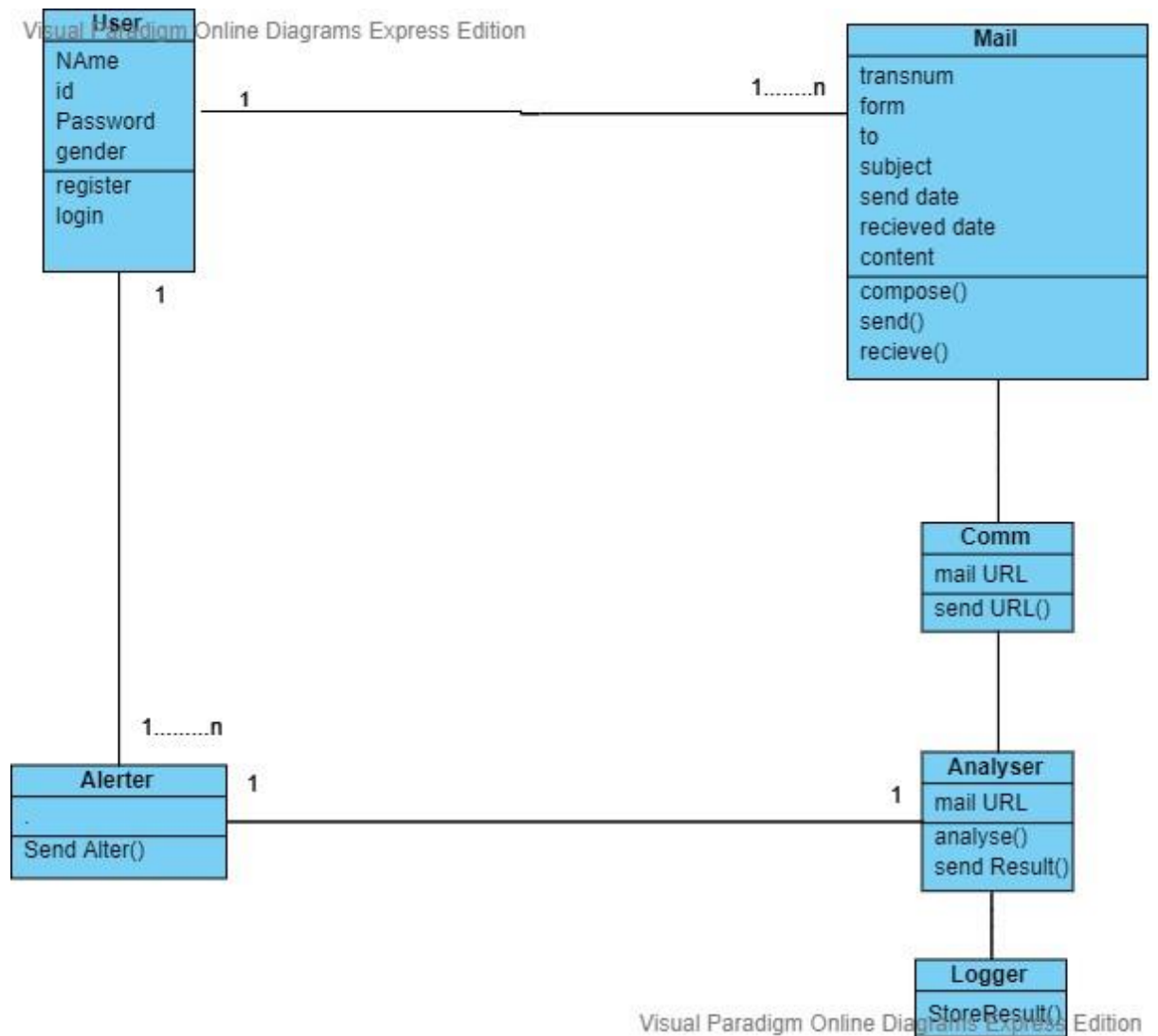
BLOCK-DIAGRAM:



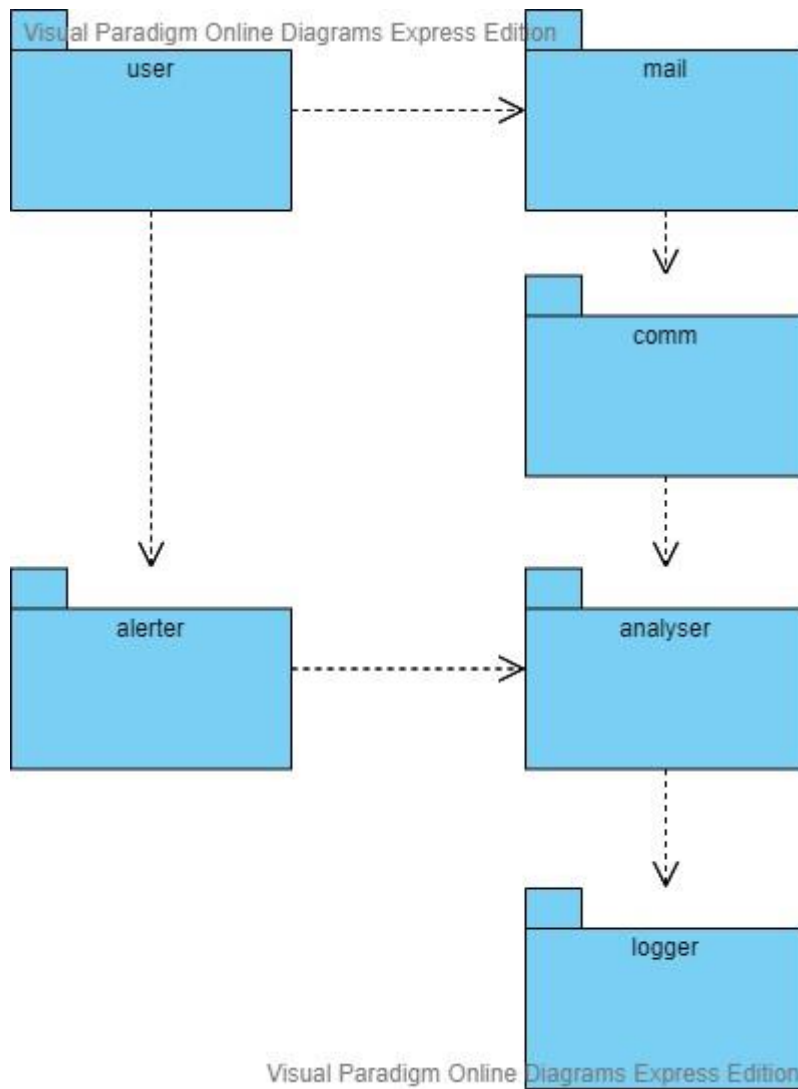
WORK-BREAKDOWN STRUCTURE:



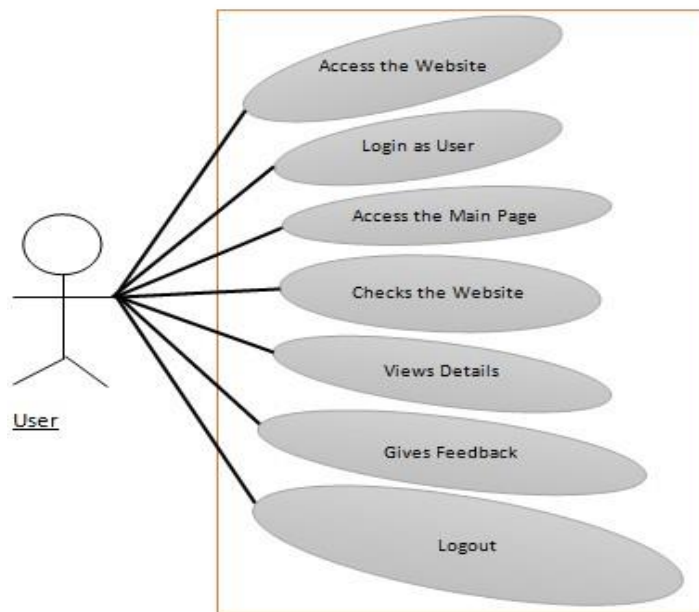
CLASS DIAGRAM-



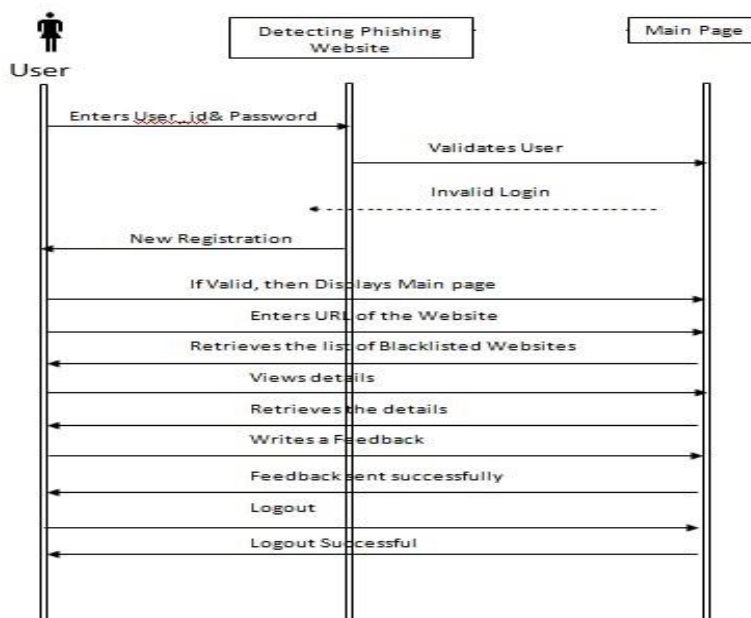
Package diagram



USE-CASE DIAGRAM:



SEQUENCE-DIAGRAM:



THE LINK GUARD ALGORITHM

```
v_link: visual link;
a_link: actual_link;
v_dns: visual DNS name;
a_dns: actual DNS name;
sender_dns: sender's DNS name.
int LinkGuard(v_link, a_link) {
1 v_dns = GetDNSName (v_link);
2 a_dns = GetDNSName (a_link);
3 if ((v_dns and a_dns are not
4 empty) and (v_dns != a_dns))
5 return PHISHING;
6 if (a_dns is dotted decimal)
7 return POSSIBLE_PHISHING;
8 if (a_link or v_link is encoded)
9 {
10 v_link2 = decode (v_link);
11 a_link2 = decode (a_link);
12 return LinkGuard(v_link2, a_link2);
13 }
14 /* analyze the domain name for
15 possible phishing */
16 if(v_dns is NULL)
17 return AnalyzeDNS (a_link);
}
```

The LinkGuard algorithm works as follows. In its main routine LinkGuard, it first extracts the DNS names from the actual and the visual links (lines 1 and 2). It then compares the actual and visual DNS names, if these names are not the same, then it is phishing of category 1 (lines 3-5). If dotted decimal IP address is directly used in actual dns, it is then a possible phishing attack of category 2 (lines 6 and 7). We will delay the discussion of how to handle possible phishing attacks later. If the actual link or the visual link is encoded

```
int AnalyzeDNS (actual link)
{
/* Analyze the actual DNS name according
to the blacklist and whitelist*/
18 if (actual_dns in blacklist)
19 return PHISHING;
20 if (actual_dns in whitelist)
21 return NOTPHISHING;
22 return PatternMatching (actual_link);
}
int PatternMatching(actual_link) {
23 if (sender_dns and actual_dns are different)
24 return POSSIBLE_PHISHING;
25 for (each item prev_dns in seed_set)
26 {
27 bv = Similarity(prev_dns, actual_link);
28 if (bv == true)
29 return POSSIBLE_PHISHING;
30 }
31 return NO_PHISHING;
}
float Similarity (str, actual_link) {
32 if (str is part of actual_link)
33 return true;
34 int maxlen = the maximum string
35 lengths of str and actual_dns;
36 int minchange = the minimum number of
37 changes needed to transform str
38 to actual_dns (or vice verse);
39 if (thresh<(maxlen-minchange)/maxlen<1)
40 return true
41 return false;
```

```
}
```

(Categories 3 and 4), we first decode the links, then recursively call LinkGuard to return a result (lines 8-13). When there is no destination information (DNS name or dotted IP address) in the visual link (category 5), LinkGuard calls AnalyzeDNS to analyze the actual dns (lines 16 and 17). LinkGuard therefore handles all the 5 categories of phishing attacks. AnalyzeDNS and the related subroutines are depicted in Fig.2. In AnalyzeDNS, if the actual dns name is contained in the blacklist, then we are sure that it is a phishing attack (lines 18 and 19). Similarly, if the actual dns is contained in the whitelist, it is therefore not a phishing attack (lines 20 and 21). If the actual dns is not contained in either whitelist or blacklist, PatternMatching is then invoked (line 22).

CODE

DETECTING OF PISHING WEBSITES AND BLACKLISTING

```
using System;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Net;
using System.IO;
using mshtml;
using System.Text.RegularExpressions;
using System.Data;
using System.Data.SqlClient;
public partial class Web : System.Web.UI.Page
{
    SqlConnection con = new SqlConnection(@" Data Source=HP;Initial Catalog=Phishing;Integrated Security=True");
    protected void Page_Load(object sender, EventArgs e)
    {
34
        // Label2.Text = "0";
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        I1.Visible = true;
        try
        {
            string URL = TextBox1.Text;
            if (TextBox1.Text != "")
            {
                I1.Attributes.Add("src", URL);
            }
            // Create a request for the URL.
            string url = TextBox1.Text;
            WebRequest request = WebRequest.Create(url);
            // If required by the server, set the credentials.
            request.Credentials = CredentialCache.DefaultCredentials;
```

```

// Get the response.
HttpWebResponse response = (HttpWebResponse)request.GetResponse();
// Display the status.
Console.WriteLine(response.StatusDescription);
// Get the stream containing content returned by the server.
Stream dataStream = response.GetResponseStream();
// Open the stream using a StreamReader for easy access.
StreamReader reader = new StreamReader(dataStream);
// Read the content.
string responseFromServer = reader.ReadToEnd();
// Display the content.
// Cleanup the streams and the response.
reader.Close();
dataStream.Close();
response.Close();
//reads the html into an html document to enable parsing
IHTMLDocument2 doc = new HTMLDocumentClass();
doc.write(new object[] { responseFromServer });
doc.close();
WebClient x = new WebClient();
string source = x.DownloadString(TextBox1.Text.ToString());
string title = Regex.Match(source, @"<title\b[^>]*>\s*(?<Title>[\s\S]*?)</title>",
RegexOptions.IgnoreCase).Groups["Title"].Value;
title1.Text = title;
//loops through each element in the document to check if it qualifies for the attributes to be set
foreach (IHTMLElement el in (IHTMLElementCollection)doc.all)
{
// check to see if all the desired attributes were found with the correct values
if (el.tagName == "META")
{
HTMLMetaElement meta = (HTMLMetaElement)el;
if (meta.name == "keywords")
{
int kcount = 0;
String[] sent = meta.content.Split(',');
foreach (string word in sent)
{
String[] words = word.Split(',');
foreach (string sword in words)
{
Console.WriteLine(sword);
}
}
kcount = kcount + 1;
}
}
// Label3.Text = Convert.ToString(kcount);
}
}
}
}
}

```

```

catch (UriFormatException)
{
String alertmsg = "Enter Proper Url Starting with http://";
ClientScript.RegisterStartupScript(this.GetType(), "myalert", "alert('" + alertmsg + "');", true);
}
string url1 = TextBox1.Text;
string s4 = "select url,title,desn from blist where url='"+TextBox1.Text+"'";
SqlDataAdapter sda4 = new SqlDataAdapter(s4, con);
DataSet ds4 = new DataSet();
sda4.Fill(ds4);
int flag = 0;
int cc = ds4.Tables[0].Rows.Count;
cc = cc - 1;
if (ds4.Tables[0].Rows.Count > 0)
{
Label9.Text = "Blacklisted Website";
Label9.ForeColor = System.Drawing.Color.Red;
Page.ClientScript.RegisterStartupScript(GetType(), "msgbox", "alert('This website is Phishing Website!!!', true);", true);
}
else
{
Label9.Text = "Not a Blacklisted Website";
Label2.Text = "0";
Label9.ForeColor = System.Drawing.Color.Black;
string ks = "select * from bwords";
SqlDataAdapter sda1 = new SqlDataAdapter(ks, con);
DataSet dss = new DataSet();
sda1.Fill(dss);
int cnt = dss.Tables[0].Rows.Count;
int words = 0;
while (cnt > 0)
{
for (int i = 0; i < cnt; i++)
{
string t1 = TextBox1.Text;
string ks1 = dss.Tables[0].Rows[i][0].ToString();
if (words == 0)
{
if (t1.Contains(ks1))
{
words = 1;
int ccc = Convert.ToInt32(Label2.Text) - 1;
Label6.Text = "The URL contains Blacklisted Words!!!";
Label6.ForeColor = System.Drawing.Color.Red;
Label6.Visible = true;
Label2.Text = ccc.ToString();
break;
}
}
}
}
}

```

```

if (words == 1)
{
int ccc = Convert.ToInt32(Label2.Text) + 1;
Label2.Text = ccc.ToString();
break;
}
}
if (words == 1)
{
break;
}
cnt--;
}
if (words == 0)
{
Label6.Text = "The URL does not contain phishing keywords!!!";
Label6.ForeColor = System.Drawing.Color.Black;
}
string s = TextBox1.Text;
int len = s.Length-5;
string s1 = s.Remove(5,len);
if (s1 == "https")
{
int c = Convert.ToInt32(Label2.Text) + 1;
Label2.Text = c.ToString();
Label7.Visible = true;
Label7.Text = "The website is secure since it uses https ";
Label7.ForeColor = System.Drawing.Color.Black;
}
else
{
int c = Convert.ToInt32(Label2.Text) - 1;
Label7.Text = "The website may not be secure since it uses http instead of https";
Label7.ForeColor = System.Drawing.Color.Red;
Label7.Visible = true;
Label2.Text = c.ToString();
}
string[] ss2 = url1.Split('.');
string chk = "select url,title,desn from chklist";
SqlDataAdapter sda = new SqlDataAdapter(chk, con);
DataSet ds = new DataSet();
sda.Fill(ds);
int flag1 = 0;
int cc1 = ds.Tables[0].Rows.Count - 1;
if (ds.Tables[0].Rows.Count > 0)
{
while (cc1 >= 0)
{
for (int i = 0; i < ss2.Length; i++)

```

```

{
ss2[i] = ss2[i].ToLower();
if (flag1 == 0)
{
if (ds.Tables[0].Rows[cc1][0].ToString().Contains(ss2[i]))
{
string sss = ds.Tables[0].Rows[cc1][2].ToString();
string tit = ds.Tables[0].Rows[cc1][1].ToString();
string s0 = title1.Text;
title1.Text = s0.Replace("\r\n", "");
if (ds.Tables[0].Rows[cc1][1].ToString() == title1.Text && ds.Tables[0].Rows[cc1][2].ToString() == Label5.Text)
{
Label8.Text = "The url,title and description is found";
Label8.ForeColor = System.Drawing.Color.Black;
flag1 = 1;
int l2 = Convert.ToInt32(Label2.Text) + 2;
Label2.Text = l2.ToString();
break;
}
}
}
if (flag1 == 0)
{
Label8.Text = "The url,title and description is not found";
Label8.ForeColor = System.Drawing.Color.Red;
break;
}
}
cc1--;
} //while
} //if
string str1 = "select * from list where url='" + TextBox1.Text + "'";
SqlDataAdapter da = new SqlDataAdapter(str1, con);
DataSet sds = new DataSet();
da.Fill(sds);
if (sds.Tables[0].Rows.Count > 0)
{
}
else
{
if (Convert.ToInt32(Label2.Text) < 0)
{
con.Open();
string s6 = "insert into list values('" + TextBox1.Text + "', '" + Label2.Text + "', 'Yes')";
SqlCommand cmd1 = new SqlCommand(s6, con);
cmd1.ExecuteNonQuery();
con.Close();
con.Open();
string s5 = "insert into blist values('" + TextBox1.Text + "', '" + title1.Text + "', '" + Label5.Text + "')";

```

```

SqlCommand cmd = new SqlCommand(s5, con);
cmd.ExecuteNonQuery();
con.Close();
Page.ClientScript.RegisterStartupScript(GetType(), "msgbox", "alert('This Website may be Phishing Website')", true);
}
else
{
con.Open();
string s6 = "insert into list values('" + TextBox1.Text + "','"+ Label2.Text + "','No')";
SqlCommand cmd1 = new SqlCommand(s6, con);
cmd1.ExecuteNonQuery();
con.Close();
}
}
} //else
/* if (Convert.ToInt32(Label2.Text)<= 0)
{
Panel1.Visible = true;
Panel2.Visible = false;
}
else
{
Panel1.Visible = false;
Panel2.Visible = true;
} */
}
}

```

LOGIN CODE :

```

<%- include('partials/header') %>
<div class="ui inverted segment">
<div class="ui inverted secondary pointing menu">
<div id="logo">
<div class="header item"><i class="hotjar icon"></i><a href="/">Phishing Site</a></div></div>
<a href="/" class="item">
Home
</a>
<a href="/about" class="item">
About
</a>
<div class="right menu">
<a href="/signup" class="item">
<div class="ui primary button">Signup</div>
</a>

</div>
</div>
</div>
<link rel="stylesheet" type="text/css" href="login.css">
<form action="/login" method="POST">

```



```

    <h1>LOGIN</h1>
<div id="user">
    <i class="user icon"></i>
    <input type="Email" name="username" placeholder="USERNAME">
</div>

<div id="pass">
    <i class="key icon"></i>
    <input type="Password" name="password" placeholder="Password">
</div>
<button>LOGIN</button>
</div><br>
<p>ToRegister<a href="/signup">SIGNUP</a></p>
</form>

```

APP CODE :

```

var express      = require("express");
var mongoose     = require("mongoose");
var methodoverride = require("method-override");
var passport     = require("passport");
var bodyparser   = require("body-parser");
var localstrategy = require("passport-local");
var passportlocalmongoose = require("passport-local-mongoose");
var User         = require("./models/user");

mongoose.connect("mongodb://localhost/MSWEB",{useNewUrlParser:true,useUnifiedTopology: true});
var app=express();
app.use(express.static("sheets"));
app.use(express.static("public"));
app.set("view engine","ejs");
app.use(bodyparser.urlencoded({extended: true}));
app.use(methodoverride("_method"));

app.use(require("express-session")({
  secret:"MS IS ONE TO MAKE HISTORY",
  resave:false,
  saveUninitialized:false
}));

app.use(passport.initialize());
app.use(passport.session());
passport.use(new localstrategy(User.authenticate()));
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());

app.get("/about",function(req, res) {
  res.render("about");
});
//////ROUTES

```

```

app.get("/",function(req,res){
    res.render("home");
});
app.get("/homepage",isLoggedIn,function(req,res){
    res.render("homepage");
});

//AUTH ROUTES
app.get("/signup",function(req, res) {
    res.render("signup");
});
app.post("/signup",function(req,res){

    req.body.username
    req.body.password
    User.register(new User({username:req.body.username, firstname : req.body.firstname ,lastname
: req.body.lastname,phone:req.body.phone,Birthday:req.body.Birthday,gender:req.body.gender}),r
eq.body.password,function(err,User){
    if(err){
        console.log("err");
        return res.render('signup');
    }
    passport.authenticate("local")(req,res,function(){
        res.render("homepage");
    });

});

});

//LOGIN ROUTES
app.get("/login",function(req, res) {
    res.render("login");
});
app.post("/login",passport.authenticate("local",{

    successRedirect:"/homepage",
    failureRedirect:"/login",
}),function(req,res){

});

//logout route
app.get('/logout', function(req, res, next) {
  if (req.session) {
    // delete session object
    req.session.destroy(function(err) {
      if(err) {
        return next(err);

```

```

    } else {
        return res.redirect('/login');
    }
});
}
});

var blogSchema = new mongoose.Schema({
  Title: String,
  image: String,
  body : String,
  created :{type: Date, default:Date.now}
});
var blog = mongoose.model("blog", blogSchema)

//Restful Routes
app.get("/",isLoggedIn,function(req,res){
  res.redirect("/blogs");
});
//Index Route
app.get("/blogs",isLoggedIn,function(req,res){
  blog.find({}, function(err, blogs){
    if(err){
      console.log("Error");
    }else{
      res.render("post",{blogs:blogs});
    }
  });
});
});

//New Route
app.get("/blogs/new",isLoggedIn,function(req, res) {
  res.render("new");
});
//Create Route
app.post("/blogs",isLoggedIn,function(req,res){
  blog.create(req.body.blog,function(err, newblog){
    if(err){
      res.render("new");
    }else{
      res.redirect("/blogs");
    }
  });
});
//SHOW ROUTE
app.get("/blogs/:id",isLoggedIn, function(req,res){
  blog.findById(req.params.id,function(err, foundblog){
    if(err){
      res.redirect("/blogs");
    }else{
      res.render("show",{blog: foundblog});
    }
  });
});

```

```

    }
  });
});
//Edit Route
app.get("/blogs/:id/edit",isLoggedIn,function(req, res) {
  blog.findById(req.params.id,function(err, foundblog){
    if(err){
      res.redirect("/blogs");
    }else{
      res.render("edit",{blog :foundblog})
    }
  });
});
//Update Route
app.put("/blogs/:id",isLoggedIn,function(req,res){
  blog.findByIdAndUpdate(req.params.id,req.body.blog, function(err ,updateblog){
    if(err){
      res.redirect("/blogs");

    } else{
      res.redirect("/blogs/" + req.params.id);
    }
  });
});
//Delete Route
app.delete("/blogs/:id", isLoggedIn,function(req,res){
  blog.findByIdAndRemove(req.params.id, function(err){
    if(err){
      res.redirect("/blogs");
    } else{
      res.redirect("/blogs");
    }
  });
});

function isLoggedIn(req,res,next){
  if(req.isAuthenticated()){
    return next();
  }
  res.redirect("/login");
}

app.listen(process.env.PORT,process.env.IP,function(){
  console.log("UR SERVER IS READY!!!");
});

```

SIGNUP CODE:

```
<%- include('partials/header') %>
<div class="ui inverted segment">
<div class="ui inverted secondary pointing menu">
  <div id="logo">
    <div class="header item"><i class="hotjar icon"></i><a href="/">Phishing Site</a></div></div>
    <a href="/" class="item">
      Home
    </a>
    <a href="/about" class="item">
      About
    </a>
    <div class="right menu">
      <a href="/login" class="item">
        Login
      </a>
    </div>
  </div>
</div>
</div>

<link rel="stylesheet" type="text/css" href="signup.css">
<form action="/signup" method="POST">
  <input type="text" name="firstname" placeholder="Firstname" required
  oninvalid="this.setCustomValidity('U forgot to enter ur FIRST NAME')"
  oninput="this.setCustomValidity('')"><br>
  <input type="text" name="lastname" placeholder="Lastname" required
  oninvalid="this.setCustomValidity('U forgot to enter ur LAST NAME')"
  oninput="this.setCustomValidity('')"><br>
  <strong>GENDER</strong><br>
  <label>Male</label>
  <input type="radio" name="gender" value="Male"required>
  <label>female</label>
  <input type="radio" name="gender" value="Female"required>
  <label>Others</label>
  <input type="radio" name="gender" value="Others"required><br>

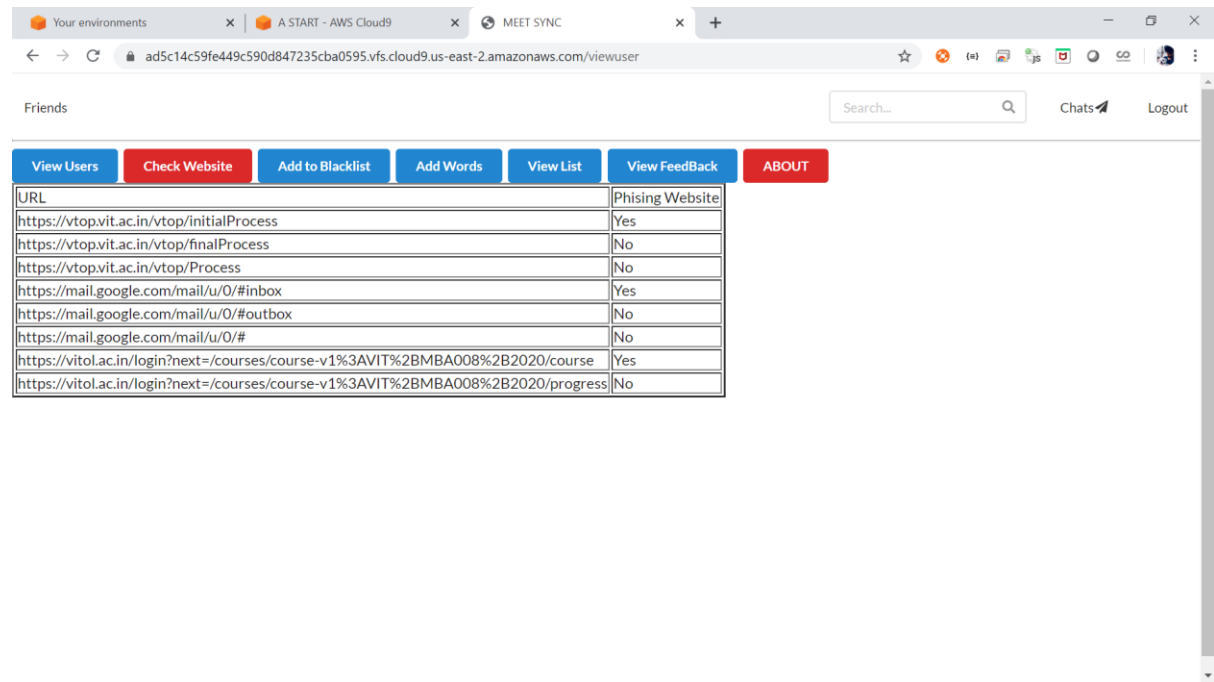
  <input type="Email" name="username" placeholder="Username" required
  oninvalid="this.setCustomValidity(' Enter a Valid one BABE!!')"
  oninput="this.setCustomValidity('')"><br>
  <input type="password" name="password" placeholder="Password" pattern=".{8,}" title="Require
  Atleast 8 charachters" required
  oninvalid="this.setCustomValidity('Require atleast 8 characters')"
  oninput="this.setCustomValidity('')"><br>

  <input type="text" name="phone" placeholder="Mobile Number" pattern="[+][9][1][6-9]{1}[0-
  9]{9}" title="Enter valid one" required
  oninvalid="this.setCustomValidity('Dont fool us...Try a Valid phone number')"
  oninput="this.setCustomValidity('')"><br>
```

```
<input type="date" name="Birthday" placeholder="date-of-birth" min="1950-01-01" max="2013-12-12" required><br>
```

```
<input type="checkbox" required> <label>I agree to the Terms and Conditions</label><br>
<button type="Submit">SUBMIT</button><br>
<div id="I">Already registered..?<a href="/login">to login</a></div>
</form>
```

TEST CASES :-



URL	Phising Website
https://vtop.vit.ac.in/vtop/initialProcess	Yes
https://vtop.vit.ac.in/vtop/finalProcess	No
https://vtop.vit.ac.in/vtop/Process	No
https://mail.google.com/mail/u/0/#inbox	Yes
https://mail.google.com/mail/u/0/#outbox	No
https://mail.google.com/mail/u/0/#	No
https://vit.ac.in/login?next=/courses/course-v1%3AVIT%2BMBA008%2B2020/course	Yes
https://vit.ac.in/login?next=/courses/course-v1%3AVIT%2BMBA008%2B2020/progress	No

RESULTS AND SCREENSHOTS :-

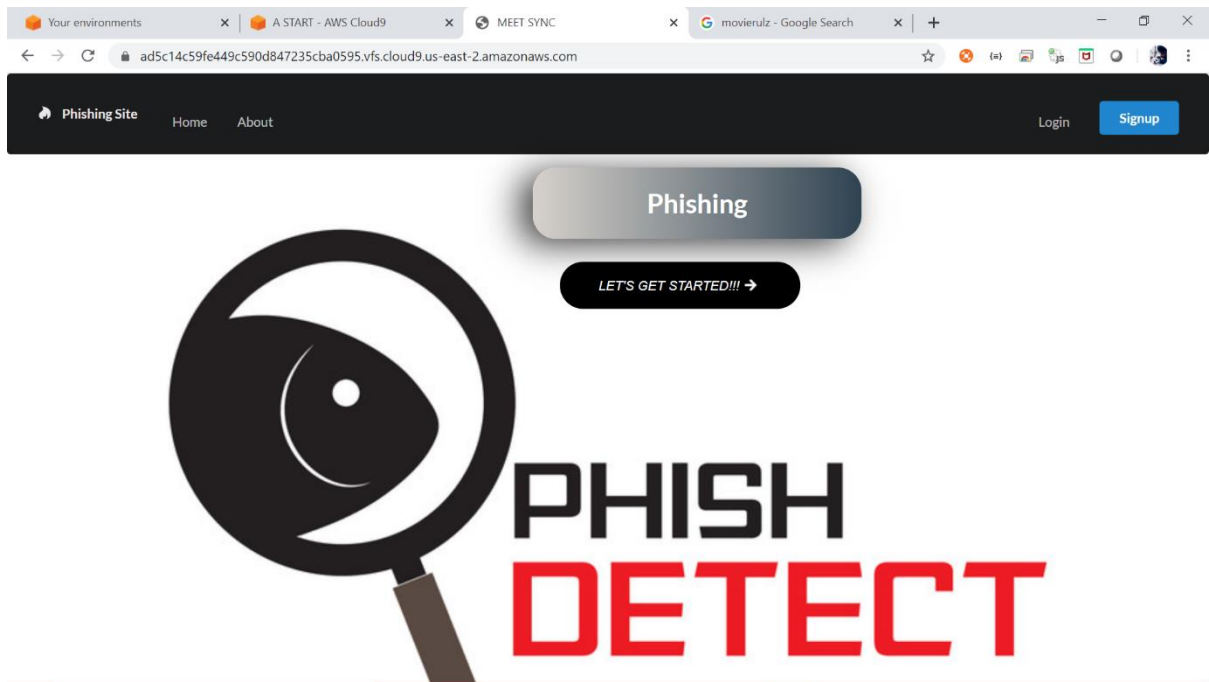


Fig1. HOMEPAGE

A screenshot of the "Signup" page of the "Phishing Site". The browser's address bar shows the URL "ad5c14c59fe449c590d847235cba0595.vfs.cloud9.us-east-2.amazonaws.com/signup". The site header is identical to the homepage. The main content area features the same magnifying glass icon and "PHISH DETECT" text. Overlaid on this is a white registration form with the following fields: "Firstname", "Lastname", "GENDER" (with radio buttons for "Male", "female", and "Others"), "Username", "Password", "Mobile Number", and a date field "dd-mm-yyyy" with a calendar icon. Below these fields is a checkbox labeled "I agree to the Terms and Conditions" and a black "SUBMIT" button.

FIG-2 SIGNUP PAGE

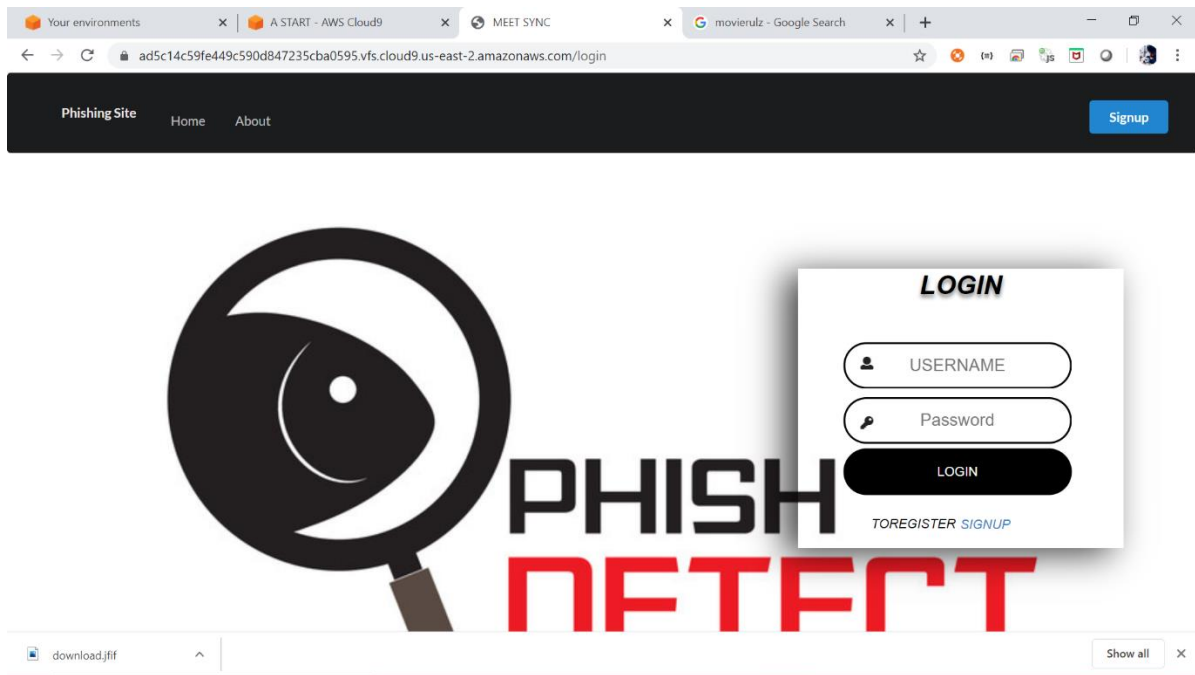


FIG-3- LOGIN PAGE

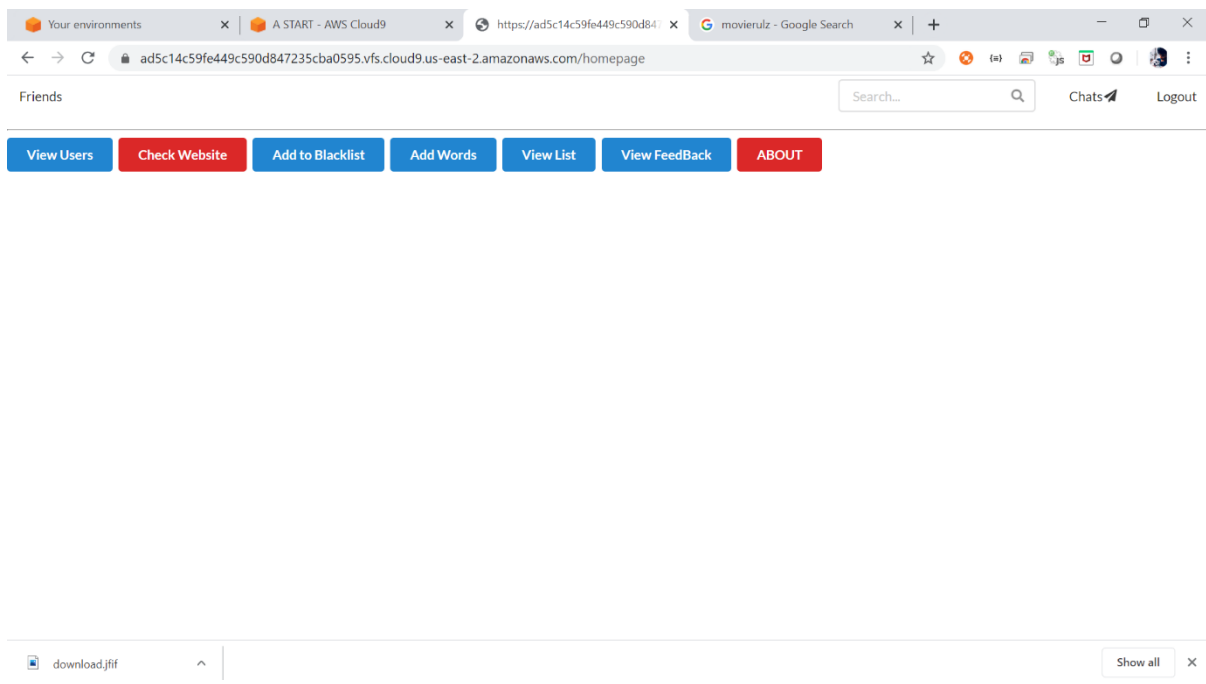
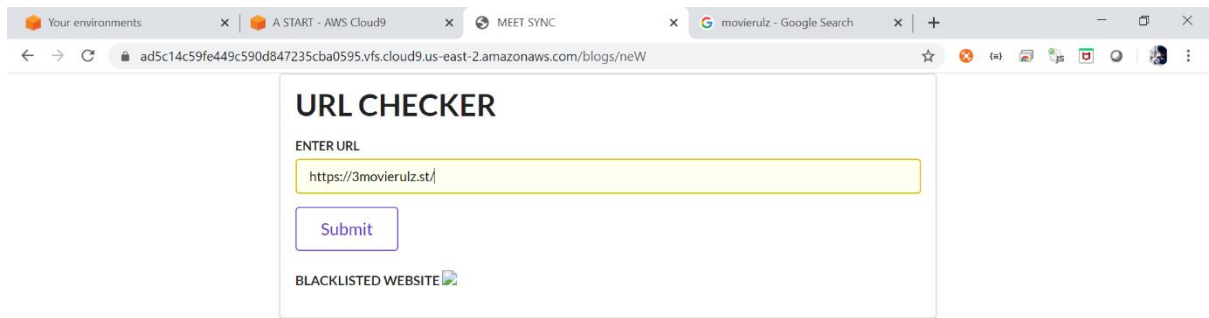


FIG-4 _ LOGIN HOME SCREEN



DONE BY MS



FIG-5_ URL DETECTOR

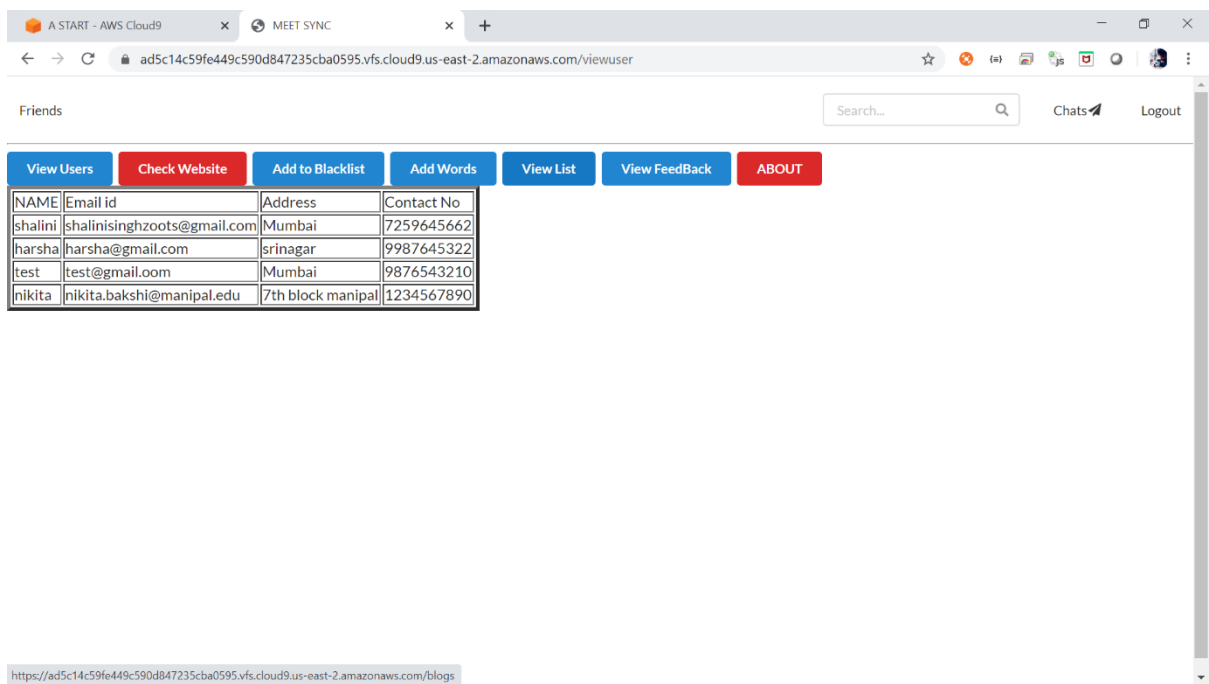


FIG-6_ USERS

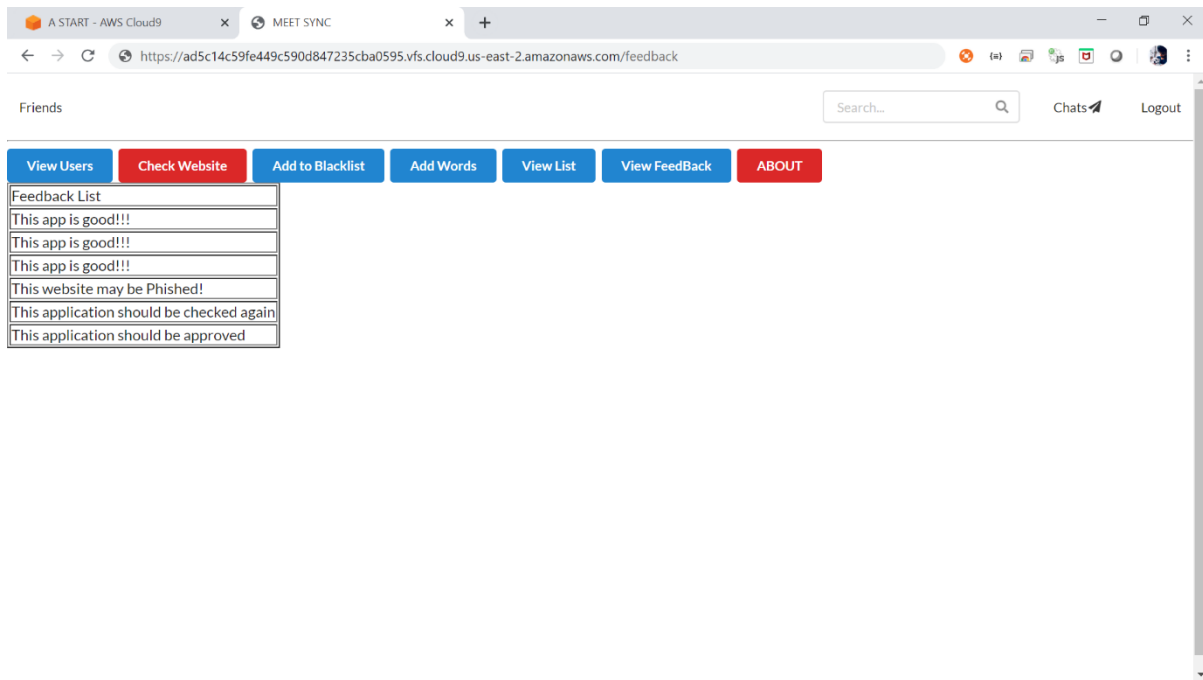


FIG-7_ FEEDBACK

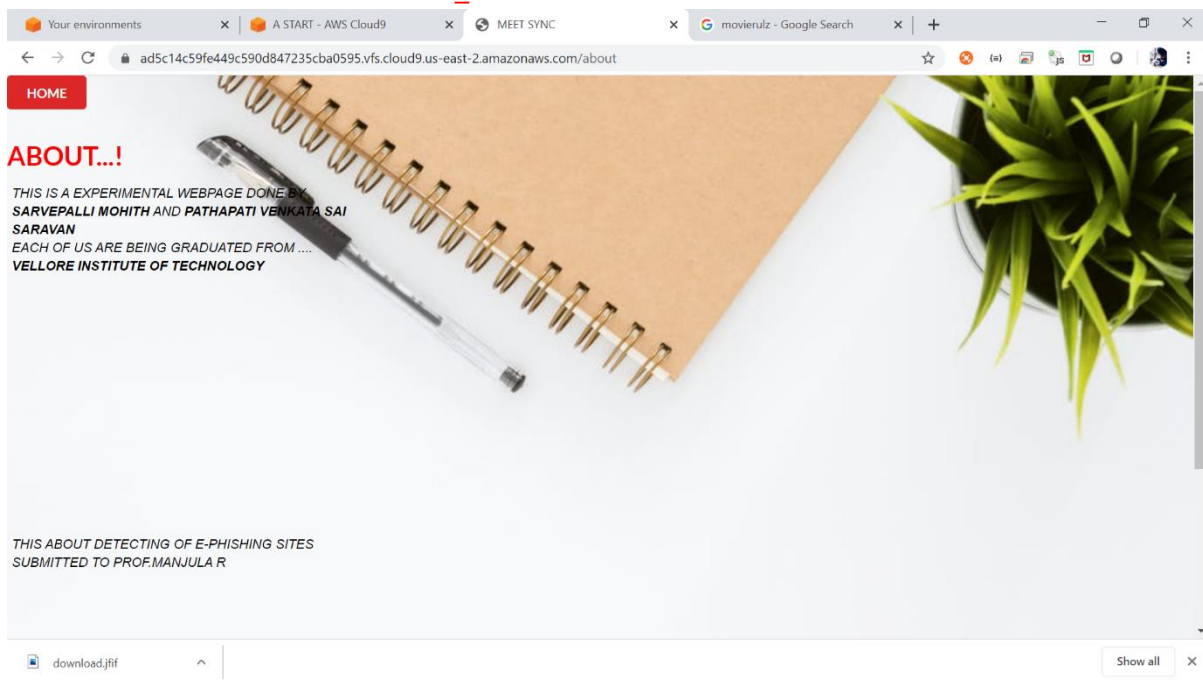


FIG-8_ ABOUT

ROLE IN SOCIETY

1. Economic Feasibility This system can be used by the E-commerce enterprise in order to carry out the whole transaction process securely. This system will increase the productivity and profitability of the E-commerce enterprise. This will provide economic benefits. It includes quantification and identification of all the benefits expected.

2. Operational Feasibility This system is more reliable, maintainable, affordable and producible. These are the parameters which are considered during design and development of this project.

During design and development phase of this project there was appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. 8

3. [Technical Feasibility](#) The back end of this project is SQL server which stores details which is related to this project. There are basic requirement of hardware to run this application. This system is developed in .Net Framework using C#. This application will be online so this application can be accessed by using any device like (Personal Computers, laptop).

FUTURE WORK AND CONCLUSION

The e-banking phishing website model showed the significance importance of the phishing website two criteria's (URL & Domain Identity) and (Security & Encryption) in the final phishing detection rate result. The model showed the correlation or relationship between some of their characteristics like the conflict of using SSL certificate with the abnormal URL request as showed from the fuzzy data mining classification and association rule algorithms. Our phishing model also showed the insignificant trivial influence of the 'Page Style & content' criteria along with 'Social Human Factor' criteria in the final rate result of detecting phishing e-banking websites, taking into consideration its characteristic association with each other like Spelling errors and Public generic salutation. Nevertheless, it is worth noting that our phishing model proved and confirmed that even if some of the e-banking phishing website characteristics or layers are not very clear or not definite, the website can still be phishy especially when other phishing characteristics or layers are obvious. On the other hand even if some of the e-banking phishing website characteristics or layers are noticed or observed, it does not mean at all that the website is phishy, but it can be safe and secured especially when other phishing characteristics or layers are not detectable. Our first goal was to determine whether we could find any golden nuggets in the e-banking phishing website archive data using classification algorithms. In this, major rules discovered were inserted into the fuzzy rule engine to help giving exact phishing rate output. A major issue in using data mining algorithms is the preparation of the feature sets to be used. Finding the "right" feature set is a difficult problem and requires some intuition regarding the goal of data mining exercise. We are not convinced that we have used the best feature sets and we think that there is more work to be done in this area. Moreover, there are number of emerging technologies that could greatly assist phishing classification that we have not considered. However, we believe that using features such as those presented here can significantly help with detecting this class of e-banking phishing websites.

References

- [1] M. Aburrous and K. Dahal, "Intelligent phishing detection system for e-banking using fuzzy data mining." Jan. 2010.
- [2] T. Moore and R. Clayton, "An empirical analysis of the current state of phishing attack and defence," vol. 1, Jan. 2007.
- [3] B. Adida, S. Hohenberger, and R. Rivest, "Lightweight encryption for email,," USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI) (ICTer), Mar. 2005.

[4] B. Liu and W. Hsu, "Integrating classification and association rule mining," Proceedings of the Fourth International Conference on KnowledgeDiscovery and Data Mining (KDD-98, Plenary Presentation), Feb. 2008.

[5] L. Nguyen, "A user modeling system for adaptive learning," International Conference on Advances in ICT for Emerging Regions (ICTer), May 2009