PROJECT REPORT ON

"PHISHING DETECTION- USING MACHINE LEARNING TECHNIQUES"

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

MASTER OF SCIENCE

in

COMPUTER SCIENCE

By

Sagar Patel

(0859196)

spatel21@lakeheadu.ca

Supervisor

Dr. Ruizhong Wei

Department of Computer Science

ABSTRACT

A phishing technique was described in detail in a paper and presentation delivered to the 1987 International HP Users Group. Ever since the concept of phishing has been introduced it's growing so fast. The problem is we do not have enough resources to detect this phishing website on a real time. Phishing attempts have grown 65% in the last year and it's still growing so fast because the average cost of a phishing attack for mid-size companies $1.6 million. The technique for detection for phishing website has been proposed and project has been implemented in this called 'PHISHING DETECTION THROUGH MACHINE LEARNING'. In this project I have emphasized on the detection through the URL also called 'DECEPTIVE PHISHING', because it is the most common way people visit the websites and links. The main aim for this project is to detect the malicious website and detect the location of the server. On top of that twenty different techniques have been used to analyze the URL type which decides if the URL is phishing or not. In the second part of the project another type of phishing has been introduced. It is called 'SPEAR PHISHING'; in this technique I have used the email header to detect the email path and some security protocols to make sure that the email we get is safe or not. In the third section I have noted some common type of phishing through Wikipedia. It is very useful for the people who do not have any prior knowledge in this area. So it is mandatory for those people to have some basic idea of phishing and its type along with how to get rid of it using this website.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning about the people who made it possible because "Success is the abstract of hard work & perseverance, but steadfast of all is encouraging guidance". So, I take this opportunity to acknowledge all those whose continuous guidance and encouragement served as a beacon light & crowned my effort, thus leading to successful completion of this work.

I take immense pleasure in expressing my deep and sincere gratitude to my supervisor, **Dr. Ruizhong Wei**, Department of Computer Science, Lakehead University for his irreproachable guidance, monitoring and constant encouragement throughout the course. His availability and creative suggestions were above and beyond what I expected. His patience, understanding, encouragement and personal guidance have proven invaluable in the preparation of this work. I appreciate his detailed discussions during my time at Lakehead University, and, for deepening my understanding of my chosen field. It has been a pleasure working under him.

My gratitude is also extended to the faculty members of Department of Computer Science for their instruction during my study. Coming from a different country, it is very difficult and can get overwhelming sometimes. I am grateful to Lakehead University for providing me such an environment to reach my goals and to prosper in the corporate world.

I would also like to thank all my friends, parents who stood behind me like strong pillars of support and never failed to present their immaculate suggestions and thus helped me in giving shape to this project as it is today.

Table of content

1. **Introduction…….…….....6**
2. **Literature Review…….…….....6**
3. **Scope of the project...…….....8**

3.1 Existing system VS Proposed System…….……9

4. **Overview of phishing…….……10**
5. **Practical use of phishing detection website…….……11**
6. **System architecture……....12**

6.1 Unified modeling language diagram……....12

6.0.1 Use case diagram……...13

6.0.2 Block machine diagram……14

6.0.3 Activity diagram……15

7. **System requirements…….....16**
8. **Project architecture…….....17**
9. **Project source code...……..28**
10. **Results……..50**
11. **Conclusion and Future work……59**
12. **References……....60**

LIST OF FIGURES

Index	Description	Page Number
1	Use-case Diagram	13
2	State-machine Diagram	14
3	Activity Diagram	15
4	Front End of URL Detection-1	17
5	Front End of URL Detection-1	18
6	Logistic regression result	50
7	Random Forest result	51
8	Support vector machine result	52
9	Variable Importance chart	52
10	Process of how to start a server	53
11	How to run Index.py file	53
12	Result of URL search	54
13	Features of URL detection results	55
14	Result of email header analyzer	55
15	Sample email header	56
16	Result of red colored link	56
17	Screenshot of wiki info page	57
18	Screenshot of wiki info page	58
19	Screenshot of wiki info page	59

1. Introduction

In this modern world everyone is accessing the internet for different purpose. As the internet grew; there are some problems like stealing data, hacking and fake web pages came into being. To get rid of these phishing websites 'phishing website detection website' is very useful. It helps users to understand what phishing is and how to stay safe while surfing the internet, because people use their personal information like credentials and bank account details.

This project has three main parts. In the first part it has 'URL detection' method. In this part what it does is when we access to the webpage, we don't know that the web page we are accessing is safe or not. So, in this first part when you copy the URL and search it tells you weather the URL is phishing or not. In the second part 'Mail phishing' it tells user to copy the email header and paste in to the given header to analyze it. Then it displays the result. In the last part called 'Wiki' it tells you about the basic types of phishing and gives the basic idea of what is phishing for the new user who doesn't have any prior knowledge. So, it is very useful nowadays because we have to do most of the things on the internet, so it should be safe and this project achieve this goal.

2. Literature Review

1) Anti-phishing solutions can be positioned at different levels of attack flow where most researchers are focusing on client-side solutions which turn to add more processing overhead at the client side and lead to losing the trust and satisfaction of the users. In this paper they introduced the anti- phishing method called 'Wide scope and fast website phishing detection using URLs lexical features. Nowadays many organizations make centralized protection of spam filtering. This paper proposes a system which can be integrated into such process to increase the detection performance in a real time. The simulation results of the proposed system showed a phishing URLs detection accuracy with 93% and provided online process of a single URL in average time of 0.12 second. They have used the below formula to calculate the accuracy.

$$Accuracy = \frac{N_{L\to L} + N_{P\to P}}{N_{L\to L} + N_{L\to P} + N_{P\to L} + N_{P\to P}}$$

Accuracy: defined as the percentage of the correct classification
over all attempts of classification. [1]

2) In the research paper 'Using domain top similarity features in machine learning' they proposed a web base similarity to add features in detection. they first experiment 200 web data in which 100 webs were phishing, the result in terms of f-measure was 0.9250 with 7.50% of error rate. The system uses 8 different features to detect the phishing web. They used the vector system to differentiate the good and bad web page and calculate the error rate. The result part includes that the result from the SVM and Random forest were 8.50% but the AdaBoost had 10.00% accuracy. The last result section tells that It is easy to implement and can achieved with 19.50% error rate and 0.8312 f-measure. [2]

3) The approach for the given research paper 'Automatic Detection of Phishing Target from Phishing Webpage' is to be clustering the webpage set consisting of its all associated webpages and the given webpage itself. We first find its associated webpages, and then explore their relationships to the given webpage as their features for clustering. The methodology includes link relationship, ranking relationship, text and layout similarity and they get the result. In the result section, they had selected 8745 phishing URLs targeting at 76 well-known companies/brands from Phish Tank to test the phishing target identification accuracy of our method. The accuracy rate for the given method is 91.44%. and the false alarm rate is 3.40%. [3]

4) In this paper 'A new method of detection of phishing website' the primary focus of this paper is to put forth a model as a solution to detect phishing websites by using the URL
detection method using Random Forest algorithm. There are 3 major phases such as Parsing, Heuristic Classification of data, Performance Analysis. The methodology is depending on the formula given.

formula: IG = H(Y) − H (Y \X) = H (X) − H (X \Y)

They have used the R studio for analysis and the data has been divided into 7:3 ratio means 70% training and 30% testing. As a result, they got 95% accuracy in random forest algorithm. [4]

5) In this paper the phishing has been done through the new method called two stage model. The methodology includes the ensemble the model based on the phishing data they got from the phish tank and then the model get the high-performance rate with 95% accuracy rate with 3.9% false positive rate. The techniques calculate false positive and true negative rate and in the last model they use random sampling method. The result part is, two stage method is accuracy= 74.9, TPR=69, FPR=1.8. For two stage model without invalid data accuracy= 95, TPR=95.3, FPR= 8.5. In the conclusion out of the 30297 websites, 12550 websites have been detected with the good rate of detection with two stage model method.[5]

3. Scope of The Project

Phishing is a considerable problem which differs from the other security threats such as intrusions and Malware which are based on the technical security holes of the network systems. The weakness point of any network system is its Users. Phishing attacks are targeting these users depending on the trikes of social engineering. [6]. There are many possible ways to do phishing but unfortunately, we have some techniques available nowadays can stop spear and email phishing and therefore we need to build a system which can help us to stop phishing attacks. One real example is the Google mail phishing attack happened some years ago and many users lost their trust on Google. So, to avoid all these big threats, we need to think possible ways to avoid large scale attacks to protect users. The scope for the phishing in this project is to save maximum users on the web and the one who work on emails. The motto of the PHISHING DETECTION project is when user is about to enter any information such as personal info or bank information we recommend them to search that link or analyze the header of that email in the web detect site first and if it shows the result is legitimate then they are safe to give the information over the internet. The scopes of the project are as follows:

- To develop a system which can cover the large scope of safe surfing on the internet with using URL detection method; URL is the most common type user interact with possible viruses and attacks
- To begin the awareness of current threat and possible future fraud methods for all users which can educate them about how to stay away from attacks and spam websites and not to become them bate
- A good practical solution for big companies to keep their customer safe and win their trust and give satisfied services.
- Mail header analyzing system introduced for big organizations and people as a good filter to avoid scam happened through emails and links which lead them to fake web page.
- A very quick and user-friendly system to check and proceed with safety to save time, money, energy and peace of mind.

3.1 Existing System VS Proposed System

As we all aware that internet has been growing so fast since 1983 and it became the part of our life now. When we take the tradition method into account there were also spoofing, and scam happened to people. There are two main reason why the tradition method of phishing has not been prevalent anymore.

First, the users in the beginning had lack of knowledge about the world wide web and how to interact with safety. User had no idea what kind of danger was there using internet and what happened if they share their personal information over internet. Their purpose was just to use www as the source of information, news and entertainment. Attackers in that time were using simple methods to steal data. According to one study, 2016 report produced by security training firm PhishMe highlighted that employees continue to be vulnerable to phishing attacks, with an average response rate of approximately 20% (Computer Fraud & Security 2016, PhishMe 2016). This includes responses to both spear phishing and generic phishing emails. This report, which was based on the analysis of over 8 million simulated phishing emails, also highlighted that 67% of employees who respond to simulated phishing attacks are repeat victims and therefore likely to respond to phishing emails more than once. The continuing vulnerability of many organizations to phishing attacks has led the UK National Cyber Security Centre to recently release specific

guidance for organizations regarding how they can defend themselves from the phishing threat (NCSC, 2018a) [7].

Second thing user nowadays are more aware, and they have advance detection system. They have knowledge that they are not supposed to trust the fake emails and avoid clicking the links which seems fake and redirect to the unreliable source page. Users are savvier than ever and have idea no to trust inbound messages which request for personal information. On top of that big companies like Google and Firefox recommends their user to change their passwords frequently to keep them safe from outside threats. As a part of proposed system Phishing website has been made using Machine Learning technique which is very powerful for data field. The enormous amounts of data can be handled through ML and it helps user to stop using malicious content because the model has been trained to detect spam emails, malicious URL and messages. In addition, the new system has less response time, it gives the result in less than 3 seconds plus it can be updated using cloud technology so user don't have to worry about getting latest version every time. The organization are using two ways authentication to improve security; for example, if you open the Gmail account on a different device then it asks for a OTP (One time password) as a verification. Also new methods called QR code scanning can reduce the possibilities of getting into the trouble like when we access the WhatsApp web online it need the QR code scan from your mobile phone which is good method to protect data and it has option to display saved logins.

4. Overview of phishing

Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels. In this process the victim receives the message that looks like it has been sent by the known person or organization. In this message it has some links or viruses which tricks user into divulging personal and bank information such as password, username and account IDs. Phishing is very common among attackers because it is easier for them to trick people to make them click on the malicious link which seems legitimate than trying to break

through the computer's defense system. The malicious links within the body of the message designed to make it appear that they go to the fake organization using that fake company's logo and other legitimate content.

5. Practical use of phishing detection website

The main aim of phishing is to steal personal information and use it for some benefits. Now as a part of solution we need to detect this attack before the attacker make us fool. For example, if you get and email from the bank you have your account in saying that you must register account through the given link otherwise it will be closed. So, what most of the people do is they believe that that email is from the bank and they trust that their information will go to the bank, however the attacker spoofs the email and steal all the information. The process has five steps.

- **Planning**: attacker decides which business or organization to target and how to get email ids of the customers for that organization. They often use the mass-mailing technique.
- **Setup**: Once they know which business to spoof and what their customers are, they find a way to deliver the message with the spoof links or documents with company's logo to get trust from customers.
- **Attack**: This is the most familiar step for everyone; in this step phisher sends a phony message that seems like from reputable source.
- **Fraud**: The data or information phisher gathered; they use it to fraud like transfer money from other account or make illegal purchase.

To get rid of this fraud and spoofing emails or messages we need to develop a system like phishing detection website which helps people to stop getting into all these troubles and save their time, money and energy. We need to build a system like detection website which people can trust and check the URL link before entering their person data and credentials.

6. System Architecture

6.1 Unified modeling language diagrams

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. UML is specifically constructed through two different domains they are:

1. UML Analysis modeling focuses on the user model and structural model views of the system.
2. UML design modeling focuses on the behavioral modeling, implementation modeling and environmental model views.

UML is a way of visualizing a software program using a collection of diagrams. The key to making a UML diagram is connecting shapes that represent an object or class with other shapes to illustrate relationships and the flow of information and data. [6]

A picture is worth a thousand words, this idiom absolutely fits describing UML. Object-oriented concepts were introduced much earlier than UML. At that point of time, there were no standard methodologies to organize and consolidate the object-oriented development. It was then that UML came into picture.

6.1.0 Use-case Diagram

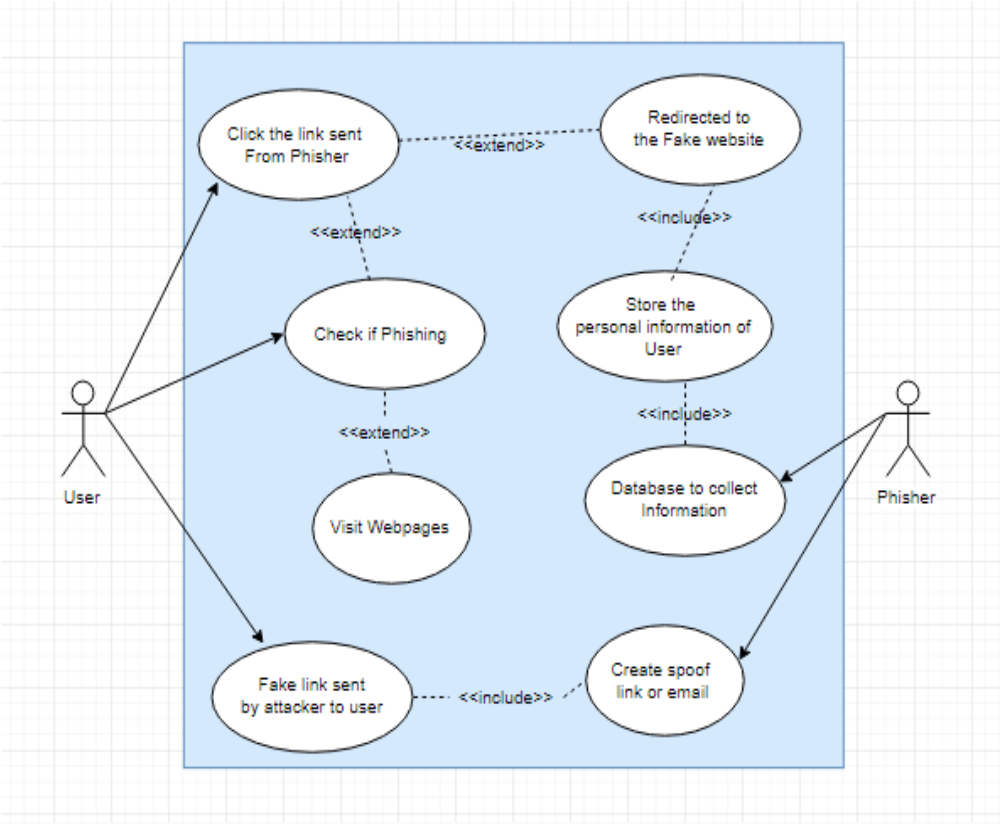


Figure 1. Use-Case Diagram

As we can see in the above Use-case diagram, there are two entities; user and phisher. User has the control over click on link sent by attacker OR user can open the mail. Also, user must check the URL or link to make sure it is not a phishing one. If it is safe to check the webpage then he/she can proceed further. On the other hand, the phisher creates a fake webpage which help to save the information for future use and make user to click on that fake email link or message.

6.2.0 State-machine Diagram

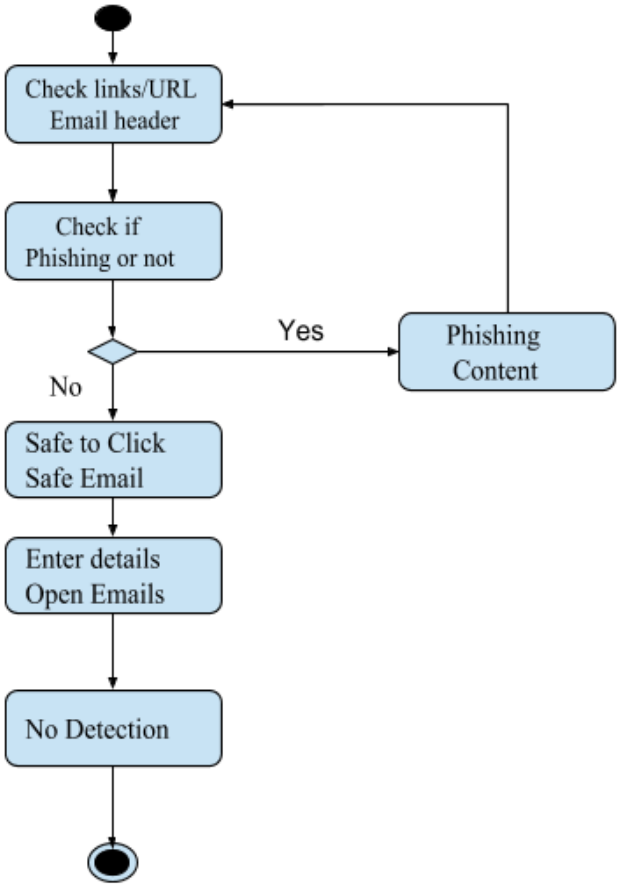


Figure 2. State-Machine Diagram

As we can see the block machine diagram is for phishing process. Although this diagram is old technique, but it can be used to explain the system better here, so we will explain based on project requirements. The first black circle means the start of the process and the end black circle with round ring meaning end of the process.

So first any user will check the URL or link of any website he/she would like to visit, then there is a condition check symbol which decides yes or no. If yes means the website is spoofed so not safe and go back again to start. In the second step is the content is legitimate then you can open an email or enter the sensitive information in web page which is safe, and process ends.

6.3.0 Activity Diagram

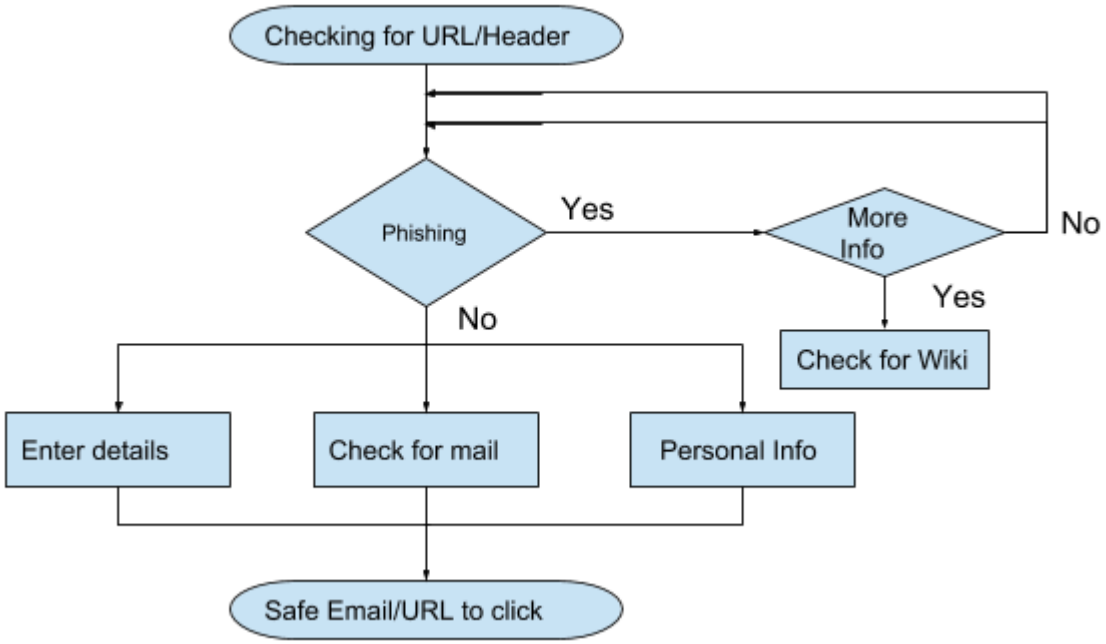


Figure 3. Activity Diagram

An activity diagram is the way to explain the flow of the process with the step by step when order of the process matters. IF we start with the first phase of user side action then user perform the header check for mail or URL check for web page he/she wants to visit before giving any sensitive info. The condition phishing check for the possible ways; if YES means not safe then if you want more info then visit the wiki page for details or else exit. If the condition NO implies the safe surfing on the internet. Enter detail, personal info and check mails are the actions can be taken by user and last stage gives the safe operation output.

7. System Requirements

The below mentioned minimum requirements are used for developing this application. We can also use the updated or other requirements depending on the user.

Software Requirements:

Operating System	Windows 8, Linux, Ubuntu
Programming Language	Python, HTML, JavaScript, CSS
Web Server	Lenovo G470
Editor	Sublime text
IDE	Spyder OR Python IDE, Eclipse

Hardware Requirements:

Processor	Intel i5
Hard Disk	500 GB
RAM	4GB

8. Project Architecture

In this project I have used many parts to build the whole system. There are two main modules and sub modules are associated with it.

1) Front End
2) Back End

FRONT END:

The front end has been made in HTML, CSS and JavaScript. The front end has three main HTML pages. The first part includes the URL search bar; second part is built to analyze the mail header and in the last part there is an information about what is Phishing and what are the different types. As we can see in the below attached screenshot that we have front page of URL detection method. In the second screenshot we can see the box to copy the email header to analyze and when we hot the search it will show the result.

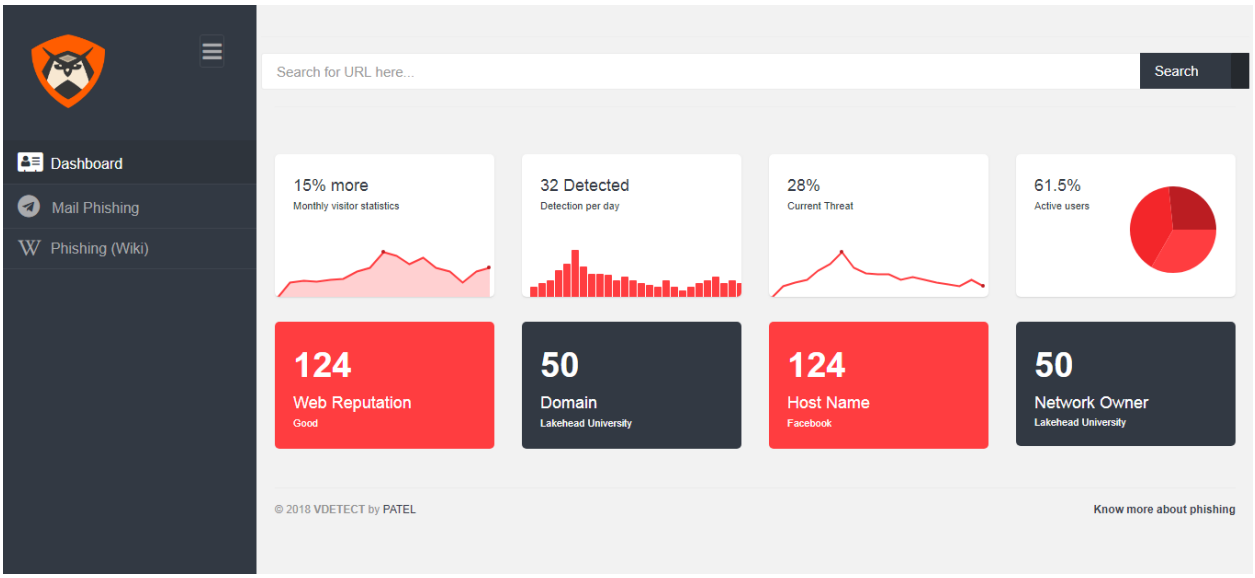


Figure 4. Front End of URL Detection-1

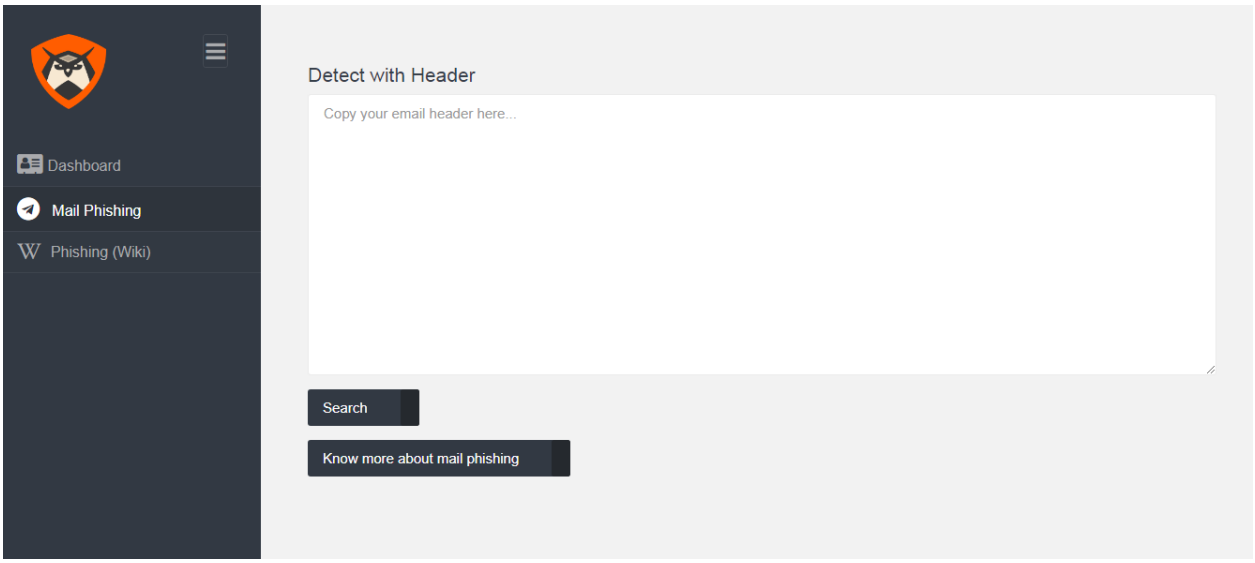


Figure 5. Front End of URL Detection-2

BACK END:

The back end has many parts and the programs have been written in the python language. The list below will help understanding the role of each coding file.

1. Dataset: the folder contains the dataset for the project. The file name is phish coop; it has been collected from the open machine learning dataset called UCI repository. the link for the dataset is: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites[8]

2. Phishing features: In this PDF file it contains the 20 different techniques to check whether the URL is phishing or not. The paper has been published by ResearchGate publication. The link for the paper used for the project is below.
    a. https://www.researchgate.net/publication/277476345_Phishing_Websites_Features[9]

3. Models: In this folder there are three python files. Logistic Regression, Random Forest and Support Vector Machine. All these three are machine learning algorithms to check which algorithm will works the best and which algorithm gives the best accuracy.

4. Chrome-driver: Chrome-driver is a WebDriver (used for automation and testing) for Chromium/Chrome. Used along with the Selenium module from Python, it covers a myriad of our objectives such as finding backlinks for a website or finding features from a mail header.

5. Index.py: This python file contains the endpoints of the Flask API which are consumed at the frontend. https://documenter.getpostman.com/view/1034072/S17oxAFE[10]

6. links.py: This python file is used to find the number of backlinks for a website using Chrome Driver and this website. https://www.semrush.com/analytics/backlinks/ [11]

7. locator.py: This python file serves the purpose of finding public server location(s) for a given domain. The 'DNS' package from python finds multiple external IP addresses for a domain and the 'geolite2' package finds the world map coordinates for a given IP address.

8. mail.py: This python file finds multiple features of a mail header using Selenium and Chrome Driver. The websites used for this purpose are -:
    a. https://toolbox.googleapps.com/apps/messageheader/ [12]
    b. https://www.iptrackeronline.com/email-header-analysis.php[13]

9. mail_Links.py: This python file analyzes the links to external websites found in a message header. For each link, we use the URL phishing module of our application to predict whether it is a legitimate or a phishing link.

10. modifiedScript.py: This python file is used to find the URL FEATURES of a given URL as described below.

URL FEATURES

Address bar features

1. Using the IP address:

If an IP address is used as an alternative of the domain name in the URL then user can be sure about someone is trying to steal the information.

Rule: IF {If the Domain Part has an IP Address → Phishing

Otherwise→ Legitimate}

2. **Long URL to hide suspicious content:**

Phishers can use long URL to hide the doubtful part in the address bar

Rule: IF $\{URL\ length < 54 \rightarrow feature$ = Legitimate

$else\ if\ URL\ length \geq 54\ and \leq 75 \rightarrow feature = Suspicious$

$otherwise \rightarrow feature$ = Phishing}

3. Using URL Shortening Services "Tiny URL":

URL shortening is a method on the "World Wide Web" in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an "HTTP Redirect" on a domain name that is short, which links to the webpage that has a long URL.

Rule: IF {Tiny URL → Phishing

Otherwise → Legitimate}

4. URL's having "@" Symbol:

Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol.

Rule: IF {URL Having @ Symbol → Phishing

Otherwise → Legitimate

5. Redirecting using "//":

The existence of "//" within the URL path means that the user will be redirected to another website. If the URL starts with "HTTP", that means the "//" should appear in the sixth position. However, if the URL employs "HTTPS" then the "//" should appear in seventh position.

Rule: IF {The Position of the Last Occurrence of "//" in the URL > 7 → Phishing
Otherwise → Legitimate

6. Adding Prefix or Suffix Separated by (-) to the Domain:

The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate web page.

Rule: IF {Domain Name Part Includes (−) Symbol → Phishing
Otherwise → Legitimate

7. Sub Domain and Multi Sub Domains:

To produce a rule for extracting this feature, we firstly have to omit the (www.) from the URL which is in fact a sub domain in itself. Then, we have to remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as "Suspicious" since it has one sub domain. However, if the dots are greater than two, it is classified as "Phishing" since it will have multiple sub domains. Otherwise, if the URL has no sub domains, we will assign "Legitimate" to the feature.

Rule: IF {Dots in Domain Part = 1 → Legitimate
Dots in Domain Part = 2 → Suspicious
Otherwise → Phishing}

8. HTTPS (Hyper Text Transfer Protocol with Secure Sockets Layer)

The existence of HTTPS is very important in giving the impression of website legitimacy, but this is clearly not enough.

Rule: IF {Use https and Issuer Is Trusted and Age of Certificate ≥ 1
Years → Legitimate
Using https and Issuer Is Not Trusted → Suspicious

Otherwise → Phishing}

9. Domain Registration Length

Since a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance. the longest fraudulent domains have been used for one year only.

Rule: IF {Domains Expires on ≤ 1 years → Phishing

Otherwise → Legitimate

10. Favicon

A favicon is a graphic image (icon) associated with a specific webpage. Many existing user agents such as graphical browsers and newsreaders show favicon as a visual reminder of the website identity in the address bar. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a Phishing attempt.

Rule: IF {Favicon Loaded From External Domain → Phishing

Otherwise → Legitimate}

11. Using Non-Standard Port:

This feature is useful in validating if a service (e.g. HTTP) is up or down on a specific server. In the aim of controlling intrusions, it is much better to merely open ports that you need. Several firewalls, Proxy and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only open the ones selected. If all ports are open, phishers can run almost any service they want and as a result, user information is threatened.

Rule: IF {Port # is of the Preferred Status → Phishing

Otherwise → Legitimate}

12. The Existence of "HTTPS" Token in the Domain Part of the URL

The phishers may add the "HTTPS" token to the domain part of a URL to trick users.

Rule: IF {Using HTTP Token in Domain Part of The URL → Phishing

Otherwise → Legitimate

Abnormal based features:

1. Request URL

Request URL examines whether the external objects contained within a webpage such as images, videos and sounds are loaded from another domain. In legitimate webpages, the webpage address and most of objects embedded within the webpage are sharing the same domain.

Rule: IF {% of Request URL < 22% → Legitimate

%of Request URL ≥ 22% and 61% → Suspicious

Otherwise → feature = Phishing}

2. URL of Anchor

An anchor is an element defined by the <a> tag. This feature is treated exactly as "Request URL".

Rule: IF {% of URL Of Anchor < 31% → $Legitimate$

% of URL Of Anchor ≥ 31% And ≤ 67% → Suspicious

Otherwise → Phishing}

3. Server Form Handler (SFH)

SFHs that contain an empty string or "about: blank" are considered doubtful because an action should be taken upon the submitted information. In addition, if the domain name in SFHs is different from the domain name of the webpage, this reveals that the webpage is suspicious because the submitted information is rarely handled by external domains.

Rule: IF {SFH is "about: blank" Or Is Empty → Phishing

SFH Refers to A Different Domain → Suspicious

Otherwise → Legitimate}

4. Submitting Information to Email

Web form allows a user to submit his personal information that is directed to a server for processing. A phisher might redirect the user's information to his personal email. To that end, a

server-side script language might be used such as “mail ()” function in PHP. One more client-side function that might be used for this purpose is the “mailto:” function.

Rule: IF {Using "mail ()" or "mailto:" Function to Submit User Information → Phishing
Otherwise → Legitimate}

5. Abnormal URL

This feature can be extracted from WHOIS database. For a legitimate website, identity is typically part of its URL.

Rule: IF {The Host Name Is Not Included in URL → Phishing
Otherwise → Legitimate}

HTML and Java based features

1. Website Forwarding

The fine line that distinguishes phishing websites from legitimate ones is how many times a website has been redirected. In our dataset, we find that legitimate websites have been redirected one-time max. On the other hand, phishing websites containing this feature have been redirected at least 4 times.

Rule: IF {#of Redirect Page ≤ 1 → Legitimate
#of Redirect Page ≥ 2 And < 4 → Suspicious
Otherwise → Phishing}

2. Status Bar Customization

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the “on Mouseover” event, and check if it makes any changes on the status bar.

Rule: IF {on Mouseover Changes Status Bar → Phishing
It Does’ Change Status Bar → Legitimate}

3. Disabling Right Click

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as "Using on Mouseover to hide the Link".

Nonetheless, for this feature, we will search for event "event. Button==2" in the webpage source code and check if the right click is disabled.

Rule: IF {Right Click Disabled → Phishing

Otherwise → Legitimate}

4. Using Pop-up Window

It is unusual to find a legitimate website asking users to submit their personal information through a pop-up window. On the other hand, this feature has been used in some legitimate websites and its main goal is to warn users about fraudulent activities or broadcast a welcome announcement, though no personal information was asked to be filled in through these pop-up windows.

Rule: IF {Pop up Window Contains Text Fields → Phishing

Otherwise → Legitimate}

5. I-Frame Redirection

I-Frame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the "frame Border" attribute which causes the browser to render a visual delineation.

Rule: IF {Using iframe → Phishing

Otherwise → Legitimate}

<u>Domain based Features</u>

1. Age of Domain

This feature can be extracted from WHOIS database (Whois 2005). Most phishing websites live for a short period of time. By reviewing our dataset, we find that the minimum age of the legitimate domain is 6 months.

Rule: IF {Age of Domain ≥ 6 months → Legitimate

Otherwise → Phishing}

2. DNS Record

For phishing websites, either the claimed identity is not recognized by the WHOIS database (Whois 2005) or no records founded for the hostname (Pan and Ding 2006). If the DNS record is empty or not found then the website is classified as "Phishing", otherwise it is classified as "Legitimate".

Rule: IF {no DNS Record for The Domain → Phishing

Otherwise → Legitimate}

3. Website Traffic

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as "Phishing". Otherwise, it is classified as "Suspicious".

Rule: IF {Website Rank < 100,000 → Legitimate

Website Rank > 100,000 → Suspicious

Otherwise → Phish

4. PageRank

PageRank is a value ranging from "0" to "1". PageRank aims to measure how important a webpage is on the Internet. The greater the PageRank value the more important the webpage. In our datasets, we find that about 95% of phishing webpages have no PageRank. Moreover, we find that the remaining 5% of phishing webpages may reach a PageRank value up to "0.2".

Rule: IF{PageRank < 0.2 → Phishing

Otherwise → Legitimate}

5. Google Index

This feature examines whether a website is in Google's index or not. When a site is indexed by Google, it is displayed on search results (Webmaster resources, 2014). Usually, phishing webpages are merely accessible for a short period and as a result, many phishing webpages may not be found on the Google index.

Rule: IF{Webpage Indexed by Google → Legitimate

Otherwise → Phishing}

6. Number of Links Pointing to Page

The number of links pointing to the webpage indicates its legitimacy level, even if some links are of the same domain (Dean, 2014). In our datasets and due to its short life span, we find that 98% of phishing dataset items have no links pointing to them. On the other hand, legitimate websites have at least 2 external links pointing to them.

Rule: IF{#Of Link Pointing to The Webpage = 0 → Phishing

#Of Link Pointing to The Webpage > 0 and ≤ 2 → Suspicious

Otherwise → Legitimate}

7. Statistical-Reports Based Feature

Several parties such as Phish Tank (Phish Tank Stats, 2010-2012), and StopBadware (StopBadware,2010-2012) formulate numerous statistical reports on phishing websites at every given period; some are monthly, and others are quarterly. In our research, we used 2 forms of the top ten statistics from PhishTank: "Top 10 Domains" and "Top 10 IPs" according to statistical-reports published in the last three years, starting in January 2010 to November 2012. Whereas for "StopBadware", we used "Top 50" IP addresses.

Rule: IF{

Host Belongs to Top Phishing IPs or Top Phishing Domains → Phishing

Otherwise → Legitimate} [14]

9. Project source code:

CODE FOR LOGISTIC REGRESSION

```
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.externals import joblib

#importing the dataset
dataset = pd.read_csv("datasets/phishcoop.csv")
dataset = dataset.drop('id', 1) #removing unwanted column
x = dataset.iloc[ : , :-1].values
y = dataset.iloc[:, -1:].values

#spliting the dataset into training set and test set
from sklearn.cross_validation import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.25, random_state =0 )

#fitting logistic regression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)

#predicting the tests set result
y_pred = classifier.predict(x_test)

#confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
#pickle file joblib
joblib.dump(classifier, 'final_models/logisticR_final.pkl')
```

CODE FOR RANDON FOREST

```
#----------------importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib

#importing the dataset
dataset = pd.read_csv("datasets/phishcoop.csv")
dataset = dataset.drop('id', 1) #removing unwanted column

x = dataset.iloc[ : , :-1].values
y = dataset.iloc[:, -1:].values

#splitting the dataset into training set and test set
from sklearn.cross_validation import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.25, random_state =0 )

#----------------applying grid search to find best performing parameters
from sklearn.model_selection import GridSearchCV
parameters = [{'n_estimators': [100, 700],
    'max_features': ['sqrt', 'log2'],
    'criterion' :['gini', 'entropy']}]

grid_search = GridSearchCV(RandomForestClassifier(),  parameters,cv =5, n_jobs= -1)
grid_search.fit(x_train, y_train)
#printing best parameters
print("Best Accuracy =" +str( grid_search.best_score_))
print("best parameters =" + str(grid_search.best_params_))
#--------------------------------------------------------------------------

#fitting RandomForest regression with best params
```

```
classifier = RandomForestClassifier(n_estimators = 100, criterion = "gini", max_features =
'log2',  random_state = 0)
classifier.fit(x_train, y_train)

#predicting the tests set result
y_pred = classifier.predict(x_test)

#confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)


#pickle file joblib
joblib.dump(classifier, 'final_models/rf_final.pkl')


#-------------Features Importance random forest
names = dataset.iloc[:,:-1].columns
importances =classifier.feature_importances_
sorted_importances = sorted(importances, reverse=True)
indices = np.argsort(-importances)
var_imp = pd.DataFrame(sorted_importances, names[indices], columns=['importance'])


#-------------plotting variable importance
plt.title("Variable Importances")
plt.barh(np.arange(len(names)), sorted_importances, height = 0.7)
plt.yticks(np.arange(len(names)), names[indices], fontsize=7)
plt.xlabel('Relative Importance')
plt.show()
```

CODE FOR SUPPORT VECTOR MACHINE

```
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
from sklearn.externals import joblib

#importing the dataset
dataset = pd.read_csv("datasets/phishcoop.csv")
dataset = dataset.drop('id', 1) #removing unwanted column
x = dataset.iloc[: , :-1].values
y = dataset.iloc[:, -1:].values

#spliting the dataset into training set and test set
from sklearn.cross_validation import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.25, random_state =0 )

#applying grid search to find best performing parameters
from sklearn.model_selection import GridSearchCV
parameters = [{'C':[1, 10, 100, 1000], 'gamma': [ 0.1, 0.2,0.3, 0.5]}]
grid_search = GridSearchCV(SVC(kernel='rbf' ),  parameters,cv =5, n_jobs= -1)
grid_search.fit(x_train, y_train)

#printing best parameters
print("Best Accurancy =" +str( grid_search.best_score_))
print("best parameters =" + str(grid_search.best_params_))

#fitting kernel SVM  with best parameters calculated

classifier = SVC(C=1000, kernel = 'rbf', gamma = 0.2 , random_state = 0)
classifier.fit(x_train, y_train)
```

```
#predicting the tests set result
y_pred = classifier.predict(x_test)

#confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

#pickle file joblib
joblib.dump(classifier, 'final_models/svm_final.pkl')
```

CODE FOR INDEX.PY

```
from sklearn.externals import joblib
import modifiedScript
from flask import Flask,request,jsonify,send_file
from validators.url import url as urlcheck
import imgkit
from locator import url_locator
from email.parser import Parser
from mail import CheckMail
from mail_links import FindLinks

app = Flask(__name__)

@app.route('/send_file')
def send_shot():
        wkhtmlpath = 'C:/Program Files/wkhtmltopdf/bin/wkhtmltoimage.exe'
        output_path = 'out.jpg'
        options={'quiet':''}
        config = imgkit.config(wkhtmltoimage=wkhtmlpath) #change this path according to your binary
installation
        url = request.args["url_ajax"]
        try:
```

```
                imgkit.from_url(url,output_path, config=config,options=options)
                return send_file(output_path)
        except Exception as e:
                try:
                        domain=url.split("//")[-1].split("/")[0].split("www.")[-1]
                        imgkit.from_url(domain,output_path, config=config,options=options)
                        return send_file(output_path)
                except Exception as e:
                        return send_file("error.jpg")
        return send_file("error.jpg")

@app.route('/send_location')
def send_location():
        try:
                output_path = 'out.jpg'
                url = request.args["url_ajax"]
                locations = url_locator.find_location(url)
                latitude=[str(i[0]) for i in locations]
                longitude=[str(i[1]) for i in locations]
                city=[str(i[2]) for i in locations]
                country=[str(i[3]) for i in locations]
                for location in locations:
                        return jsonify(
                                {
                                "latitude":latitude,
                                "longitude":longitude,
                                "city":city,
                                "country":country
                                })
                return jsonify({"latitude":"null"})
        except Exception as e:
                return jsonify({"latitude":"null"})

@app.route('/send_result', methods=['POST'])
```

```
def call_model():
    try:
        classifier = joblib.load('final_models/rf_final.pkl')
        url = request.get_json()["url_ajax"]

        #validate url
        if not urlcheck(url):
            return jsonify({"prediction":"Invalid"})

        #checking and predicting
        features = modifiedScript.main(url)
        prediction = classifier.predict(features[0])
        return jsonify({
            "prediction":str(prediction[0]),
            "url_having_ip":str(features[1][0]),
            "url_length":str(features[1][1]),
            "url_short":str(features[1][2]),
            "having_at_symbol":str(features[1][3]),
            "doubleSlash":str(features[1][4]),
            "prefix_suffix":str(features[1][5]),
            "sub_domain":str(features[1][6]),
            "SSLfinal_State":str(features[1][7]),
            "domain_registration":str(features[1][8]),
            "https_token":str(features[1][9]),
            "request_url":str(features[1][10]),
            "url_of_anchor":str(features[1][11]),
            "Links_in_tags":str(features[1][12]),
            "email_submit":str(features[1][13]),
            "redirect":str(features[1][14]),
            "iframe":str(features[1][15]),
            "age_of_domain":str(features[1][16]),
            "web_traffic":str(features[1][17]),
            "google_index":str(features[1][18]),
            "links_pointing":str(features[1][19])
```

```
            })

        except Exception as e:
            print(e)
            return jsonify({"prediction":str(e)})


@app.route('/send_header', methods=['POST'])
def mail_header():
    try:
        header = request.get_json()["header_ajax"]
        parser = Parser()
        if parser.parsestr(header)['from'] is None:
            return jsonify({"header":"Invalid"})

        features = CheckMail.run(header)
        return jsonify({
            "header":"valid",
            "time":str(features["time"]),
            "SPF":str(features["SPF:"]),
            "DKIM":str(features["DKIM:"]),
            "DMARC":str(features["DMARC:"]),
            "IP":str(features["IP"]),
            "Hostname":str(features["Hostname"]),
            "Organization":str(features["Organization"]),
            "Country":str(features["Country"]),
            "City":str(features["City"]),
            "Latitude":str(features["Latitude"]),
            "Longitude":str(features["Longitude"])
        })

    except Exception as e:
        print(e)
        return jsonify({"header":"Invalid"})
```

```
@app.route('/mail_links', methods=['POST'])
def mail_links():
	try:
		header = request.get_json()["header_ajax"]
		parser = Parser()

		links_dict = FindLinks().find(header)
		links=[]
		predictions=[]
		for key,value in links_dict.items():
			links.append(str(key))
			predictions.append(str(value))

		return jsonify({
			"links":links,
			"predictions":predictions
		})

	except Exception as e:
		print(e)
		return jsonify({"header":"Invalid"})


@app.after_request
def add_headers(response):
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers', 'Content-Type,Authorization')
    return response

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

CODE FOR MODIFIEDSCRIPT.PY

```
import re as regex
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
from urllib.parse import urlencode
import requests
import whois
import datetime
from links import backlinks


def url_having_ip(url):
    domain=url.split("//")[-1].split("/")[0].split("www.")[-1]
    try:
        socket.inet_aton(domain)
        return -1
    except:
        return 1



def url_length(url):
    length=len(url)
    if(length<54):
        return [1,length]
    elif(54<=length<=75):
        return [0,length]
    else:
        return [-1,length]
```

```
def url_short(url):
    domain=url.split("//")[-1].split("/")[0].split("www.")[-1]
    shortening_services =
['bit.do','goo.gl','ow.ly','bit.ly','tinyurl','is.gd','branch.io','buff.ly','tiny.cc','soo.gd','s2r.co','clicky.me','budurl.
com']
    for shortening_service in shortening_services:
        if shortening_service in domain:
            return -1
        else:
            return 1

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

def doubleSlash(url):
    if len(url.split("//"))>2:
        return -1
    else:
        return 1

def prefix_suffix(url):
    domain=url.split("//")[-1].split("/")[0].split("www.")[-1]
    if(domain.count('-')):
        return -1
    else:
        return 1

def sub_domain(url):
    domain=url.split("//")[-1].split("/")[0].split("www.")[-1]
    if(domain.count('.')<=1):
```

```
        return 1
    elif(domain.count('.')<=2):
        return 0
    else:
        return -1

def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()

        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth =
['Google','Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave'
,'Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']
#getting age of certificate
```

```
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear

#checking final conditions
        if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1) ):
            return 1 #legitimate
        elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
            return 0 #suspicious
        else:
            return -1 #phishing

    except Exception as e:
        return -1

def domain_registration(url):
    try:
        w = whois.whois(url)
        updated = w.updated_date
        exp = w.expiration_date
        length=0
        try:
            length = (exp[0]-updated[0]).days
        except:
            try:
                length = (exp-updated).days
            except:
                try:
                    length = (exp[0]-updated).days
                except:
                    length = (exp-updated[0]).days
```

```
        if(length<=365):
            return [1,length]
        else:
            return [-1,length]
    except:
        return [-1,0]

def favicon(url):
    #ongoing
    return 0

def port(url):
    #ongoing
    return 0

def https_token(url):
    subDomain, domain, suffix = extract(url)
    host =subDomain +'.' + domain + '.' + suffix
    if(host.count('https')): #attacker can trick by putting https in domain part
        return -1
    else:
        return 1

def request_url(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)

        linked_to_same = 0
```

```
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return [1,avg]
        if(avg>=0.22 and avg<=0.61):
            return [0,avg]
        elif(avg>0.61):
            return [-1,avg]
    except:
        return [0,-1]


def url_of_anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
```

```
        soup = BeautifulSoup(opener, 'lxml')
        anchors = soup.findAll('a', href=True)
        total = len(anchors)
        linked_to_same = 0
        avg = 0
        for anchor in anchors:
            subDomain, domain, suffix = extract(anchor['href'])
            anchorDomain = domain
            if(websiteDomain==anchorDomain or anchorDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.31):
            return [1,avg]
        elif(0.31<=avg<=0.67):
            return [0,avg]
        else:
            return [-1,avg]
    except:
        return [0,-1]

def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_meta =0
        no_of_link =0
        no_of_script =0
        anchors=0
        avg =0
        for meta in soup.find_all('meta'):
```

```
            no_of_meta = no_of_meta+1
        for link in soup.find_all('link'):
            no_of_link = no_of_link +1
        for script in soup.find_all('script'):
            no_of_script = no_of_script+1
        for anchor in soup.find_all('a'):
            anchors = anchors+1
        total = no_of_meta + no_of_link + no_of_script+anchors
        tags = no_of_meta + no_of_link + no_of_script
        if(total!=0):
            avg = tags/total

        if(avg<0.17):
            return [1,avg]
        elif(0.17<=avg<=0.81):
            return [0,avg]
        else:
            return [-1,avg]
    except:
        return [0,-1]

def sfh(url):
    #ongoing
    return 0

def email_submit(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:')):
            return -1
        else:
            return 1
    except:
```

```
        return -1

def abnormal_url(url):
    #ongoing
    return 0

def redirect(url):
    try:
        response = requests.get(url)
        if (len(response.history)<4):
            return 1
        else:
            return 0
    except:
        return 0


def on_mouseover(url):
    #ongoing
    return 0

def rightClick(url):
    #ongoing
    return 0

def popup(url):
    #ongoing
    return 0

def iframe(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('iframe')):
            return -1
```

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

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

def dns(url):
    #ongoing
    return 0

def web_traffic(url):
    try:
        domain=url.split("//")[-1].split("/")[0].split("www.")[-1]
        req = urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="+domain)
        rank = BeautifulSoup(req.read(), "xml").find("REACH")['RANK']
        if int(rank)<100000:
            return [1,rank]
```

```
        elif int(rank)>100000:
            return [0,rank]
    except Exception as e:
        print(e)
        return [-1,-1]

def page_rank(url):
    #ongoing
    return 0

def google_index(url):
    try:
        user_agent = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/48.0.2564.116 Safari/537.36'
        headers = { 'User-Agent' : user_agent}

        original_domain=url.split("//")[-1].split("/")[0].split("www.")[-1]
        query = {'q': 'info:' + original_domain}
        google = "https://www.google.com/search?" + urlencode(query)
        data = requests.get(google, headers=headers)
        soup = BeautifulSoup(str(data.content), "html.parser")
        href = soup.find(id="rso").find("div").find("div").find("a")["href"]
        found_domain = href.split("//")[-1].split("/")[0].split("www.")[-1]
        if found_domain == original_domain:
            return 1
        else:
            return -1
    except AttributeError:
        return -1


def links_pointing(url):
    return backlinks.find(url)
```

```
def statistical(url):
    #ongoing
    return 0

def main(url):
    f1= url_having_ip(url)
    f2= url_length(url)
    f3= url_short(url)
    f4= having_at_symbol(url)
    f5= doubleSlash(url)
    f6= prefix_suffix(url)
    f7= sub_domain(url)
    f8= SSLfinal_State(url)
    f9= domain_registration(url)
    f10= https_token(url)
    f11= request_url(url)
    f12= url_of_anchor(url)
    f13= Links_in_tags(url)
    f14= email_submit(url)
    f15= redirect(url)
    f16= iframe(url)
    f17= age_of_domain(url)
    f18= web_traffic(url)
    f19= google_index(url)
    f20= links_pointing(url)

    check = [[f1,
    f2[0],
    f3,
    f4,
    f5,
    f6,
    f7,
    f8,
```

```
    f9[0],
    favicon(url),
    port(url),
    f10,
    f11[0],
    f12[0],
    f13[0],
    sfh(url),f14,
    abnormal_url(url),
    f15,
    on_mouseover(url),
    rightClick(url),
    popup(url),
    f16,
    f17[0],
    dns(url),
    f18[0],
    page_rank(url),
    f19,
    f20[0],
    statistical(url)]]

    metadata = [f1,f2[1],f3,f4,f5,f6,f7,f8,f9[1],f10,f11[1],f12[1],f13[1],f14,f15,f16,f17[1],f18[1],f19,f20[1]]
    return [check,metadata]
```

10. Project Results:

In the result section there are many parts to include.

a. Logistic regression:

```
In [3]: runfile('C:/Users/Krish/Desktop/Final_Project/Back End/models/logisticRegression.py',
wdir='C:/Users/Krish/Desktop/Final_Project/Back End/models')
[[1120  129]
 [  84 1431]]
C:\Users\Krish\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Traceback (most recent call last):
```

Figure 6. Logistic regression result

Logistic regression has been used to check the possible fit of the algorithm. In this result we can see that the matrix is [1120 129] [84 1431] which means we can get possibly get the good result based on confusion matrix. The value for the matrix also tells that we can test and train the model well, so we can get the best accuracy of result.

b) Random forest:

```
C:\Users\Krish\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29:
DeprecationWarning: numpy.core.umath_tests is an internal NumPy module and should not be
imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
C:\Users\Krish\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning:
This module was deprecated in version 0.18 in favor of the model_selection module into which all
the refactored classes and functions are moved. Also note that the interface of the new CV
iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
C:\Users\Krish\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:740:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples,), for example using ravel().
  self.best_estimator_.fit(X, y, **fit_params)
Best Accurancy =0.9725003015317815
best parameters ={'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 700}
__main__:34: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
[[1181   68]
 [  19 1496]]
```

Figure 7. Random Forest result

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. As we can see in the above screenshot the result for the random forest we get is 0.97 which means the best accuracy is 97% so we can use this algorithm for our dataset. It will fit for the chosen data and will generate a good output.

c) Support vector machine:

```
C:\Users\Krish\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Best Accurancy =0.9646604752140876
best parameters ={'C': 1000, 'gamma': 0.2}
[[1185   64]
 [  26 1489]]
Traceback (most recent call last):
```

Figure 8. Support vector machine result

The result for the support vector machine is 0.96 which means the best accuracy we can get from the support vector machine is 96%. Also, we can see that the best parameter for the C is 1000 and the gamma function is 0.2 means we must split the data into 80:20 ratio for getting the good result out of the model we train. Also, the confusion matrix indicates that we can make the good train and test model for selected datasets.

d) Variable importance chart:

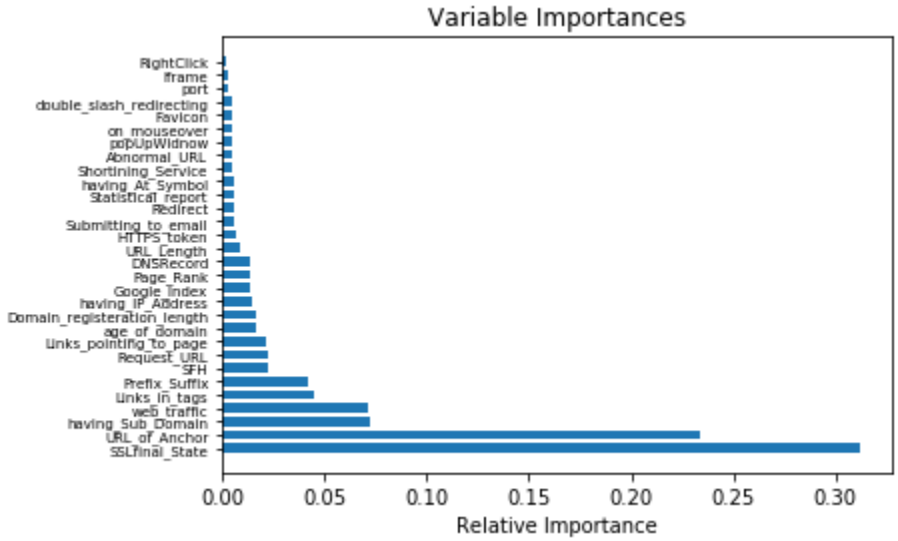


Figure 9. Variable Importance chart

The chart above is to show how much important the variables we use are. They have been acquired through the process called data cleaning; in this process we decide which variable are most important to train a final model and so it is a good way to show the importance of the variable we have used to get the results. So, it is the graph of Relative importance for all the 30 variables with the different values. As we can see the most 6 important variables are SSLfianl_state, URL of anchor, Having sub domain, Web traffic and Prefix Suffix. after that in second stage we have set of 9 variables are second important and the lase set contains the rest.

e) Starting server:

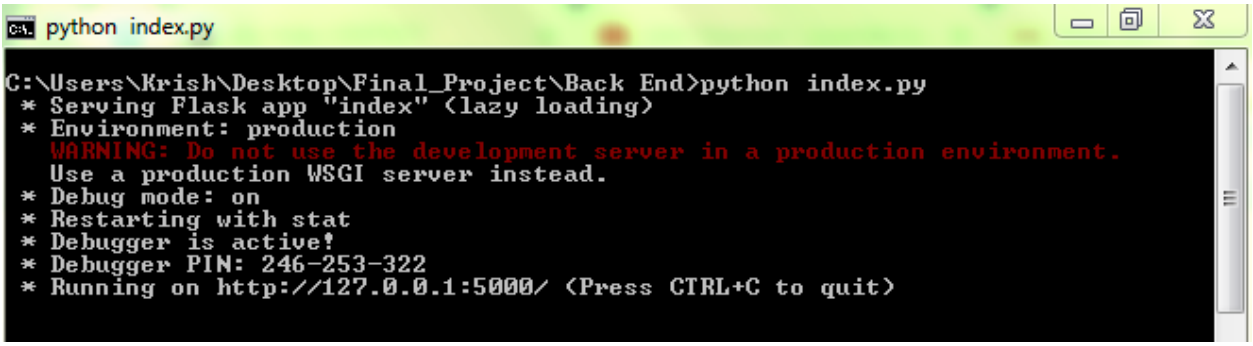


Figure 10. Process of how to start a server

To run the project, the first thing is to start the server on the local host. So, the process has to start from the back-end folder and the step one is to open the folder and then we press the SHIFT+RIGHT CLICK, we get the option in the box called “open command prompt here”.

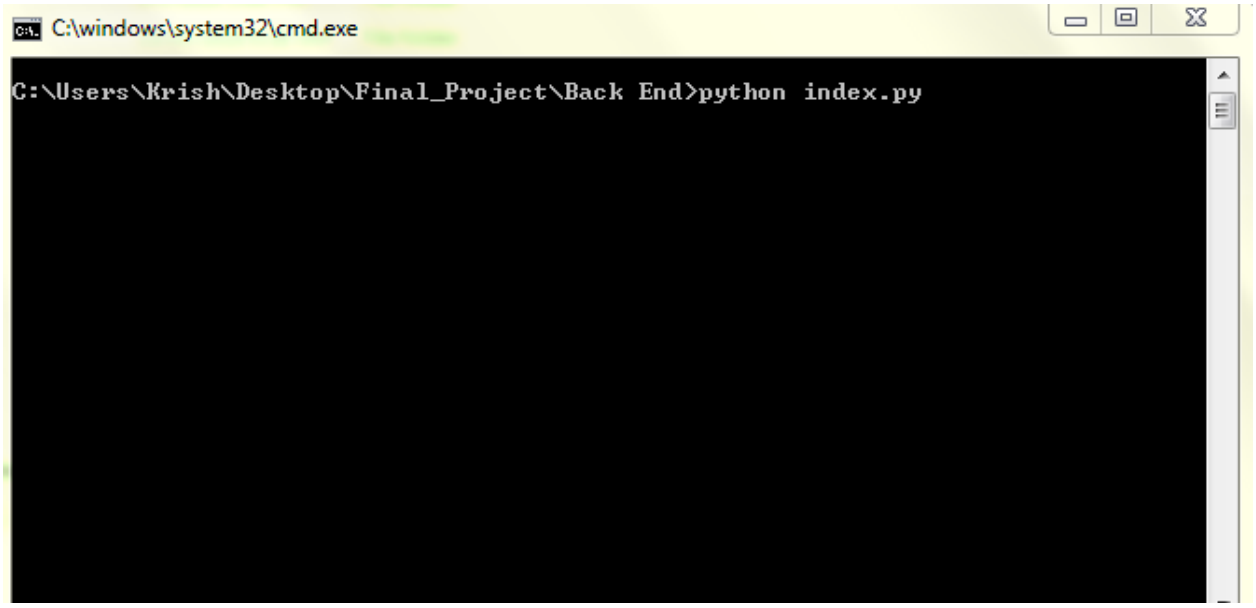


Figure 11. How to run Index.py file

Then just type python index.py as shown in the screenshot. This process run the index file and start the server on the local host and we can do the further process.

f) URL detection:

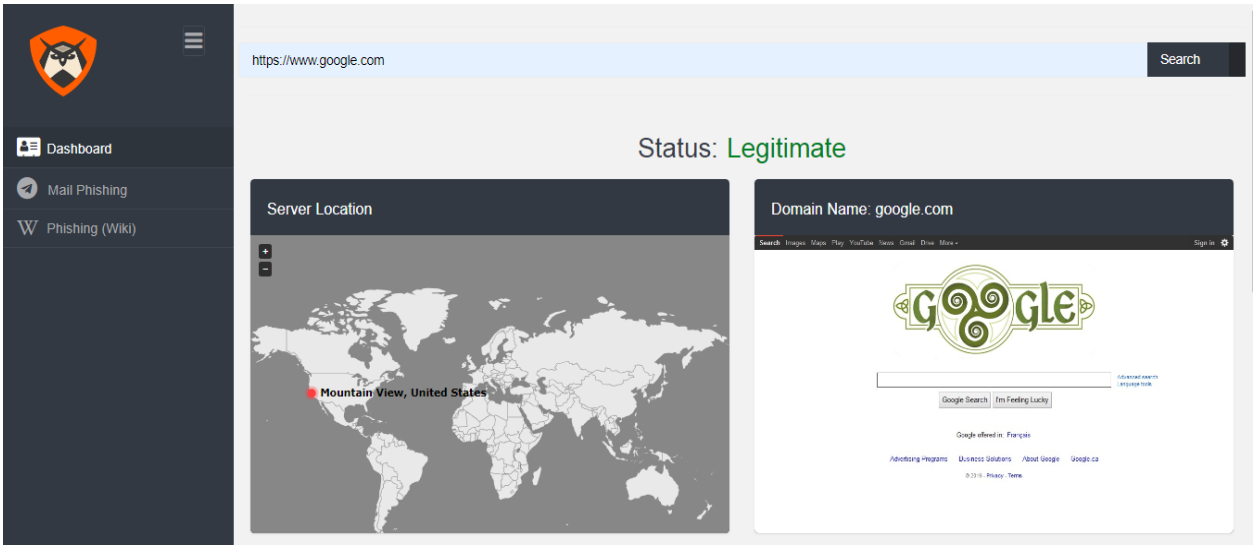


Figure 12. Result of URL search

The above screenshot is showing the first phase of the phishing website. In this picture we can see that if you try to find the google.com for example then it will give server location and the website page as a result. If the STATUS is LEGITIMATE meaning no phishing, if it is Phishing then it is not safe.

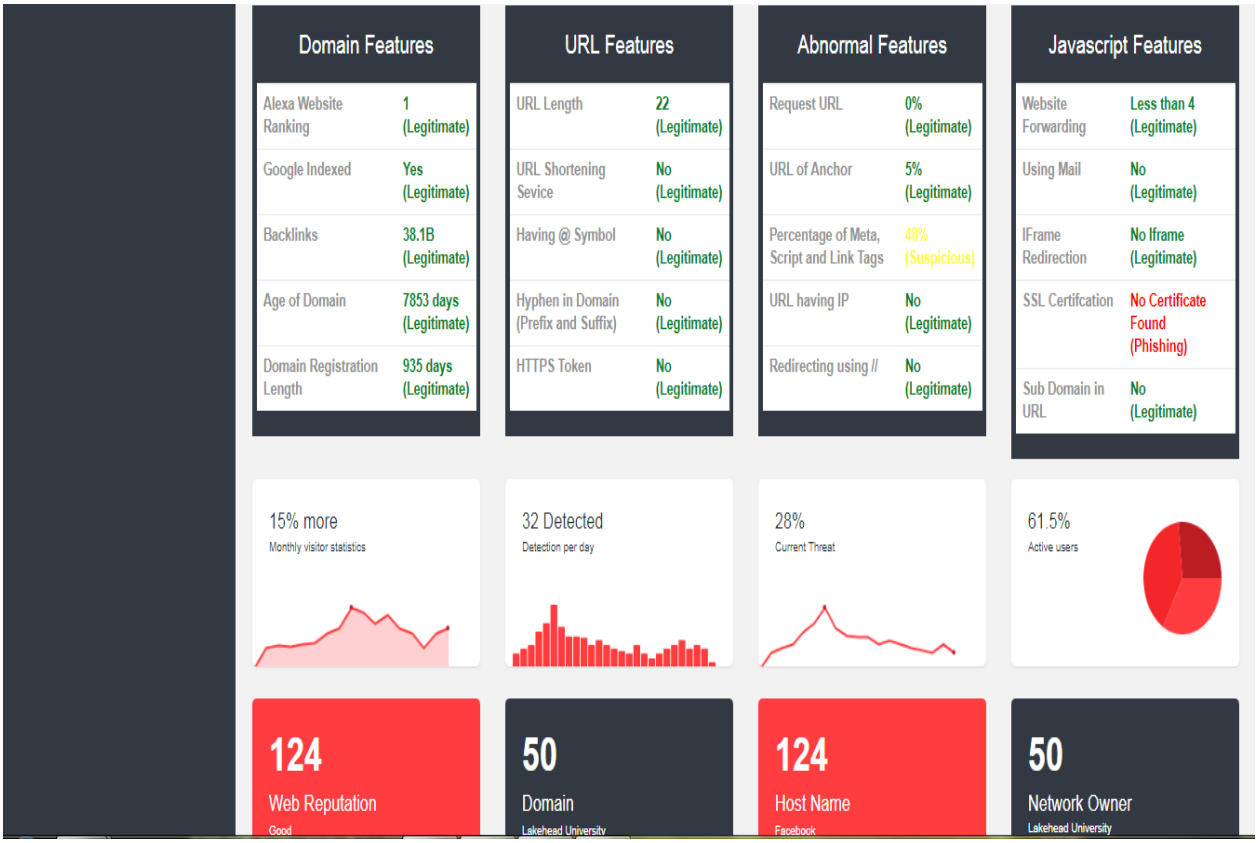


Figure 13. Features of URL detection results

The above screenshot is for explaining the twenty different features to analyze the URL link in 20 different way to detect threat. All the features are divided in four groups.

1. Domain features
2. URL features
3. Abnormal features
4. JavaScript features

The labels we can see below the features are there for future use, the person who is maintaining the website can add the other features like comments from users, adding the good ideal features according to traffic of most visited website as a comparison The first four boxes in a row can be maintained through the Json file from the admin side. The result is editable from the Json file based on the observations.

g) Mail header analyzing:

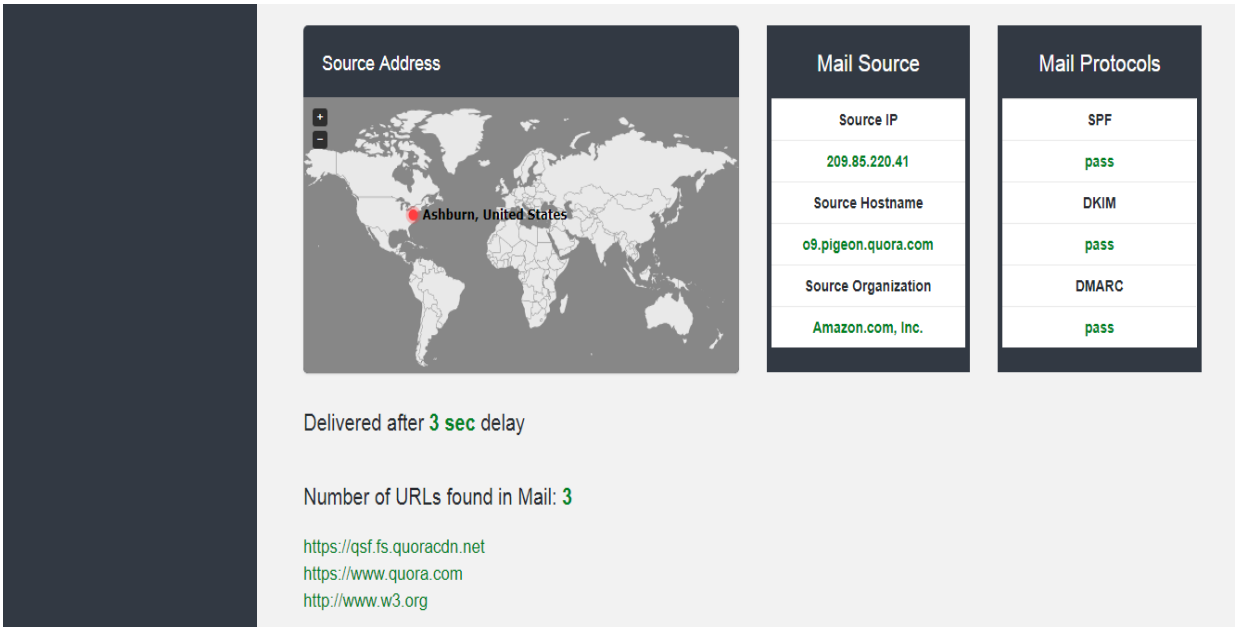


Figure 14. Result of email header analyzer

In phase 2 we have mail header analyze to check the mail is from phisher or not and it also check the links attached with the message and gives the result. First of all, when we want to check the header, you click on the email then go to options and then select ‘Show original’ as shown in the below screenshot then copy the header and paste in to the space provided in the website.

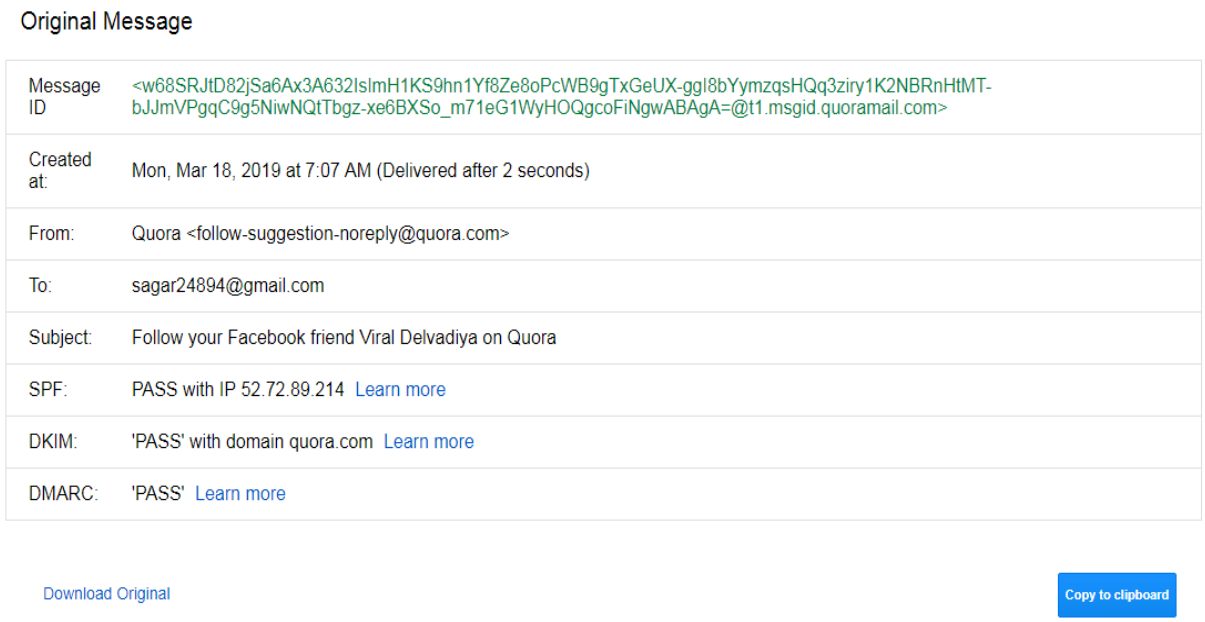


Figure 15. Sample email header

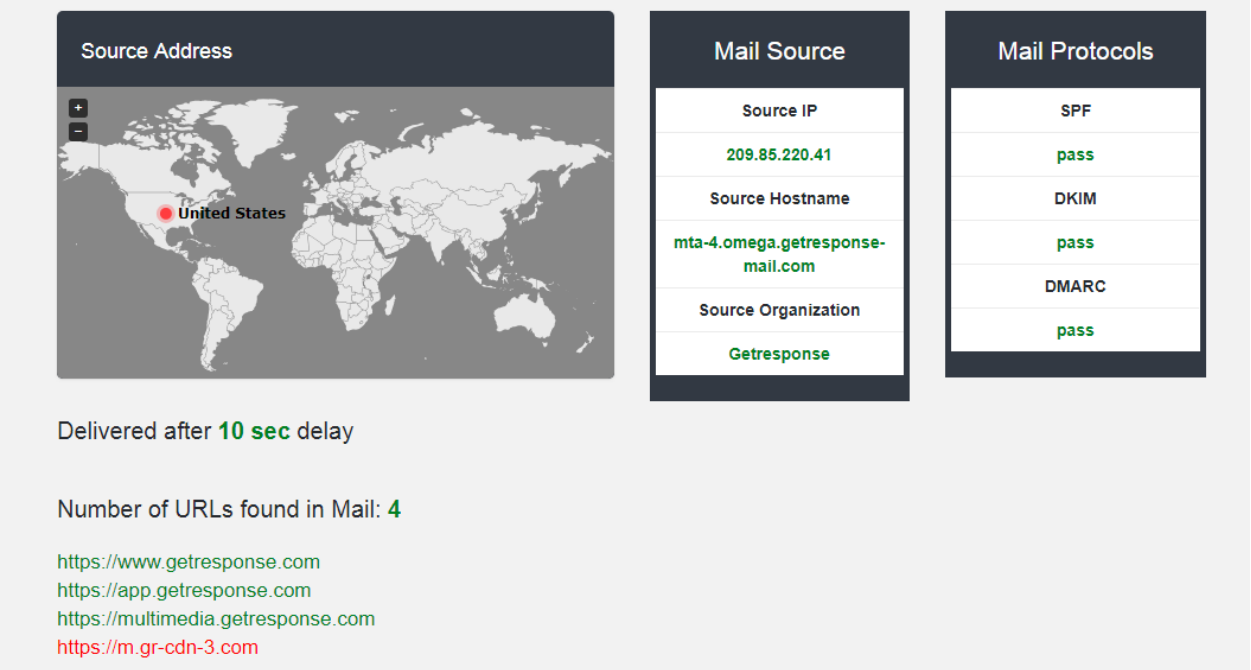


Figure 16. Result of red colored link

As a result, the website gives us the server location for an email header you want to search. On the right-hand side of the map we can see the two boxes called Mail source which

gives the details about Source IP, Source Hostname and Source organization. If they are green that means they are legitimate. If you find something in red, then it's not safe OR phishing. The next box is to check whether the mail delivery system has followed all the security protocol to prove it legitimate or not. SPF, DKIM and DMARC are three security protocol which each email must fulfil to be called safe email.

The result also gives the delay time for the packet to be delivered at the destination address in certain amount of time. If the delay time is more than 30 seconds, then it is hard to trust that kind of email. After finding all this routing information of any packet it searches for the links attacked with the message. As we can see there are total four link has been found by the system and the last link is red which means it is not safe to click on.

h) Information page about phishing:

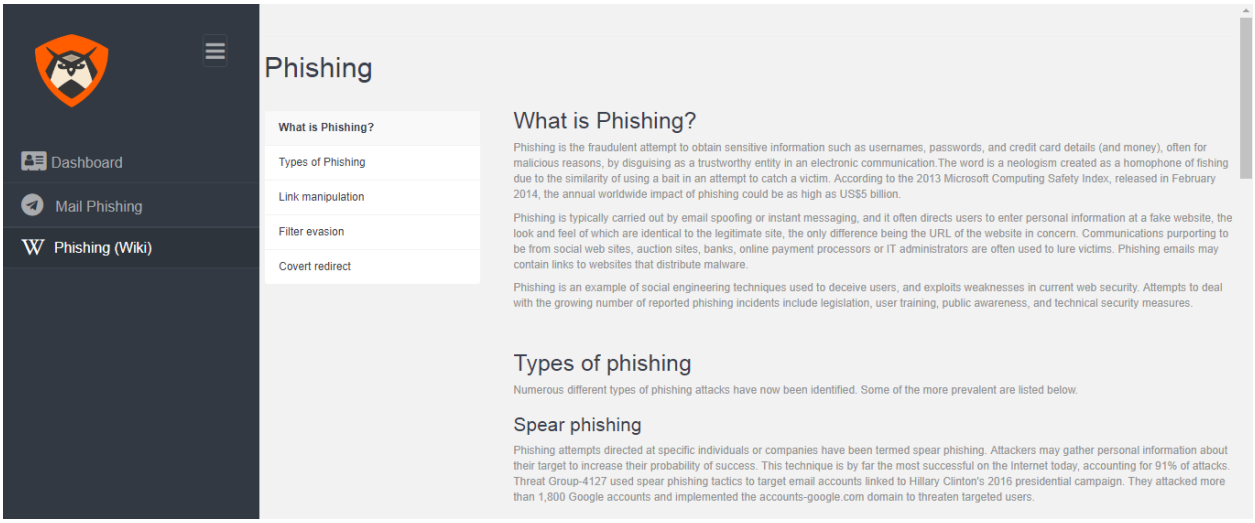
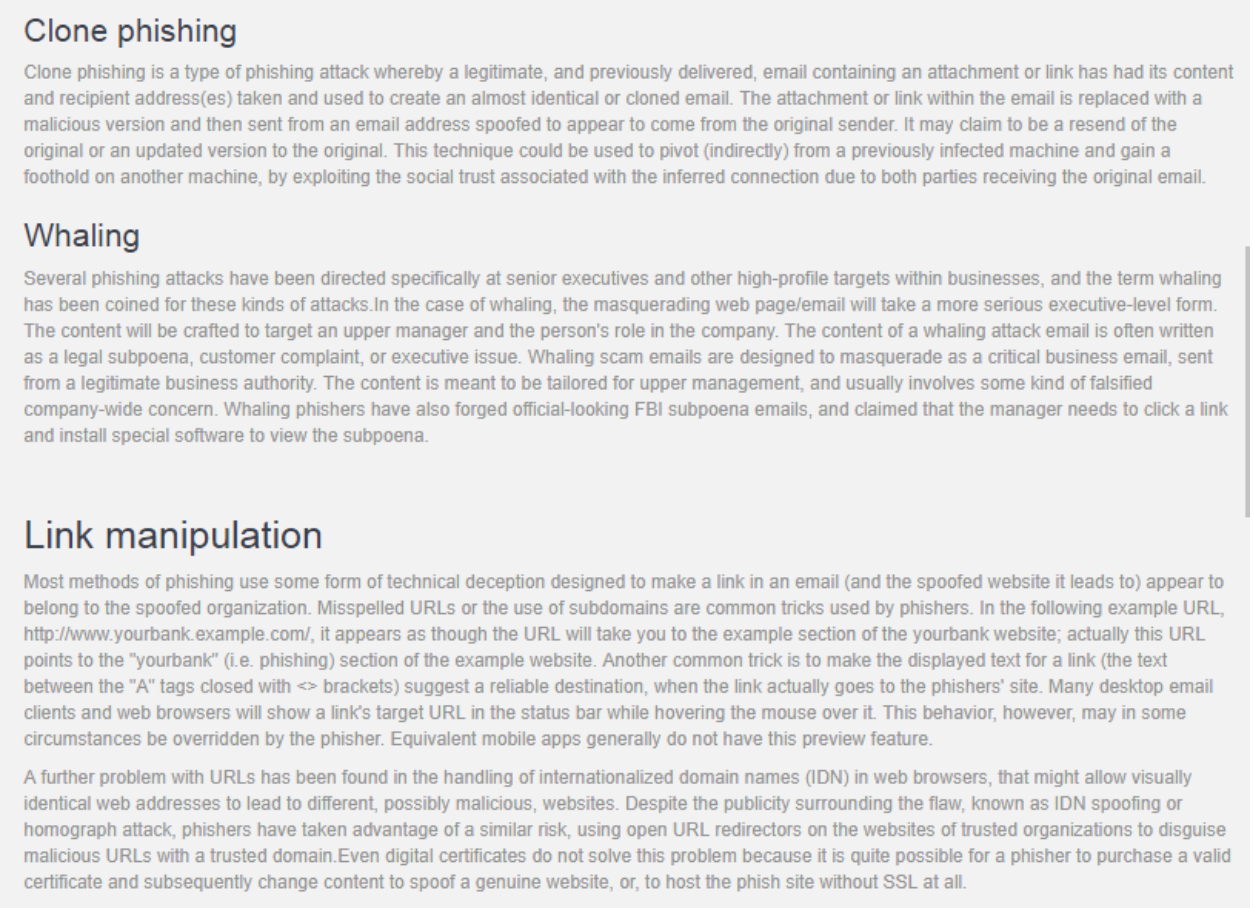


Figure 17. Screenshot of wiki info page



Clone phishing

Clone phishing is a type of phishing attack whereby a legitimate, and previously delivered, email containing an attachment or link has had its content and recipient address(es) taken and used to create an almost identical or cloned email. The attachment or link within the email is replaced with a malicious version and then sent from an email address spoofed to appear to come from the original sender. It may claim to be a resend of the original or an updated version to the original. This technique could be used to pivot (indirectly) from a previously infected machine and gain a foothold on another machine, by exploiting the social trust associated with the inferred connection due to both parties receiving the original email.

Whaling

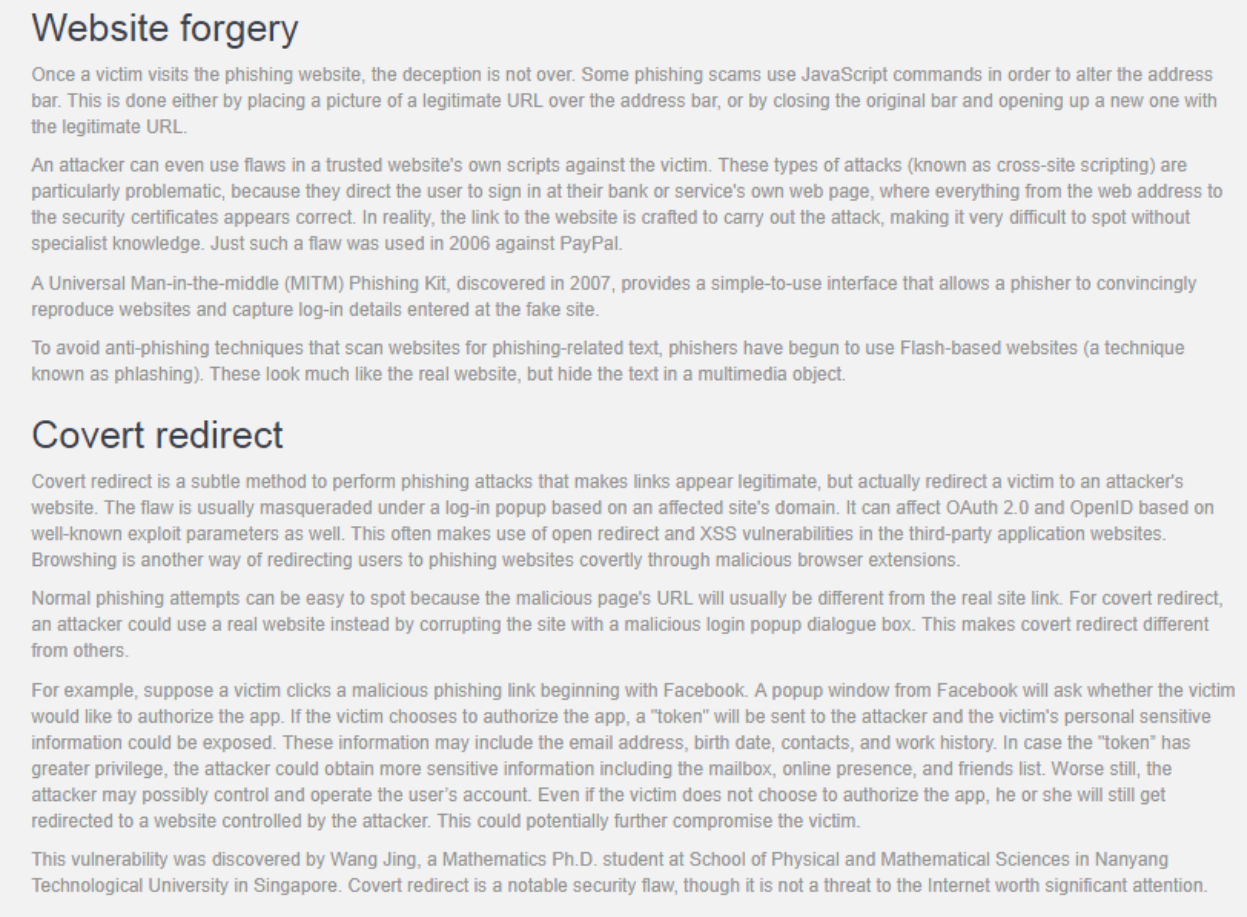
Several phishing attacks have been directed specifically at senior executives and other high-profile targets within businesses, and the term whaling has been coined for these kinds of attacks.In the case of whaling, the masquerading web page/email will take a more serious executive-level form. The content will be crafted to target an upper manager and the person's role in the company. The content of a whaling attack email is often written as a legal subpoena, customer complaint, or executive issue. Whaling scam emails are designed to masquerade as a critical business email, sent from a legitimate business authority. The content is meant to be tailored for upper management, and usually involves some kind of falsified company-wide concern. Whaling phishers have also forged official-looking FBI subpoena emails, and claimed that the manager needs to click a link and install special software to view the subpoena.

Link manipulation

Most methods of phishing use some form of technical deception designed to make a link in an email (and the spoofed website it leads to) appear to belong to the spoofed organization. Misspelled URLs or the use of subdomains are common tricks used by phishers. In the following example URL, http://www.yourbank.example.com/, it appears as though the URL will take you to the example section of the yourbank website; actually this URL points to the "yourbank" (i.e. phishing) section of the example website. Another common trick is to make the displayed text for a link (the text between the "A" tags closed with <> brackets) suggest a reliable destination, when the link actually goes to the phishers' site. Many desktop email clients and web browsers will show a link's target URL in the status bar while hovering the mouse over it. This behavior, however, may in some circumstances be overridden by the phisher. Equivalent mobile apps generally do not have this preview feature.

A further problem with URLs has been found in the handling of internationalized domain names (IDN) in web browsers, that might allow visually identical web addresses to lead to different, possibly malicious, websites. Despite the publicity surrounding the flaw, known as IDN spoofing or homograph attack, phishers have taken advantage of a similar risk, using open URL redirectors on the websites of trusted organizations to disguise malicious URLs with a trusted domain.Even digital certificates do not solve this problem because it is quite possible for a phisher to purchase a valid certificate and subsequently change content to spoof a genuine website, or, to host the phish site without SSL at all.

Figure 18. Screenshot of wiki info page



Website forgery

Once a victim visits the phishing website, the deception is not over. Some phishing scams use JavaScript commands in order to alter the address bar. This is done either by placing a picture of a legitimate URL over the address bar, or by closing the original bar and opening up a new one with the legitimate URL.

An attacker can even use flaws in a trusted website's own scripts against the victim. These types of attacks (known as cross-site scripting) are particularly problematic, because they direct the user to sign in at their bank or service's own web page, where everything from the web address to the security certificates appears correct. In reality, the link to the website is crafted to carry out the attack, making it very difficult to spot without specialist knowledge. Just such a flaw was used in 2006 against PayPal.

A Universal Man-in-the-middle (MITM) Phishing Kit, discovered in 2007, provides a simple-to-use interface that allows a phisher to convincingly reproduce websites and capture log-in details entered at the fake site.

To avoid anti-phishing techniques that scan websites for phishing-related text, phishers have begun to use Flash-based websites (a technique known as phlashing). These look much like the real website, but hide the text in a multimedia object.

Covert redirect

Covert redirect is a subtle method to perform phishing attacks that makes links appear legitimate, but actually redirect a victim to an attacker's website. The flaw is usually masqueraded under a log-in popup based on an affected site's domain. It can affect OAuth 2.0 and OpenID based on well-known exploit parameters as well. This often makes use of open redirect and XSS vulnerabilities in the third-party application websites. Browshing is another way of redirecting users to phishing websites covertly through malicious browser extensions.

Normal phishing attempts can be easy to spot because the malicious page's URL will usually be different from the real site link. For covert redirect, an attacker could use a real website instead by corrupting the site with a malicious login popup dialogue box. This makes covert redirect different from others.

For example, suppose a victim clicks a malicious phishing link beginning with Facebook. A popup window from Facebook will ask whether the victim would like to authorize the app. If the victim chooses to authorize the app, a "token" will be sent to the attacker and the victim's personal sensitive information could be exposed. These information may include the email address, birth date, contacts, and work history. In case the "token" has greater privilege, the attacker could obtain more sensitive information including the mailbox, online presence, and friends list. Worse still, the attacker may possibly control and operate the user's account. Even if the victim does not choose to authorize the app, he or she will still get redirected to a website controlled by the attacker. This could potentially further compromise the victim.

This vulnerability was discovered by Wang Jing, a Mathematics Ph.D. student at School of Physical and Mathematical Sciences in Nanyang Technological University in Singapore. Covert redirect is a notable security flaw, though it is not a threat to the Internet worth significant attention.

Figure 19. Screenshot of wiki info page

10. Conclusion and future work

As a conclusion of the project 'Phishing detection with machine learning' is a new concept. There are twenty different methods have been used to carry out the result which is fast, quick and reliable for all the users. Also, the model used for training and testing gives the best accuracy of 97% for the random forest algorithm for machine learning. The methods used for the mail header analyzer is more advanced and gives the result according the path email header had visited, and the protocols needs to be followed. So, it is very safe method of detection of spoof OR phishing content on the internet.

The future work includes admin side. The current website is open source and we can improve it with some advance features. The first one is to make it access by cloud to everyone and when people try to access the web it will automatically ask for permission to track the search

and suspicious search will be blocked. On top of that the Google-chrome extension can be made to get the on-hand safety just like antivirus. The second method introduces the admin page. This is the page where a centralized person has the control over the phishing website and admin has a right to maintain or edit the content. last but not the least, if people have some doubts then they will have the facility of live chat over web page. Also, there are some tags shown in the current website which aware user about what is an ideal website has; for example, page rank, index number, host name etc. There could be one notification center which tells the real time news about what is happening around the globe. To conclude there will be some new and good method to prevent phishing threat in upcoming future and we will have advanced way to prevent.

11. References:

1. Y. Daeef, R. B. Ahmad, Y. Yacob and N. Y. Phing, "Wide scope and fast websites phishing detection using URLs lexical features," 2016 3rd International Conference on Electronic Design (ICED), Phuket, 2016, pp. 410-415. doi: 10.1109/ICED.2016.7804679
2. N. Sanglerdsinlapachai and A. Rungsawang, "Using Domain Top-page Similarity Feature in Machine Learning-Based Web Phishing Detection," 2010 Third International Conference on Knowledge Discovery and Data Mining, Phuket, 2010, pp. 187-190. doi: 10.1109/WKDD.2010.108
3. S. Parekh, D. Parikh, S. Kotak and P. S. Sankhe, "A New Method for Detection of Phishing Websites: URL Detection," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2018, pp. 949-952. doi: 10.1109/ICICCT.2018.8473085
4. G. Liu, B. Qiu and L. Wenyin, "Automatic Detection of Phishing Target from Phishing Webpage," 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 4153-4156. doi: 10.1109/ICPR.2010.1010
5. G. Liu, B. Qiu and L. Wenyin, "Automatic Detection of Phishing Target from Phishing Webpage," 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 4153-4156.doi: 10.1109/ICPR.2010.1010
6. J. Li and S. Wang, "PhishBox: An Approach for Phishing Validation and Detection," 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl

Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, 2017, pp. 557-564. doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.101

Bibliography

1. https://en.wikipedia.org/wiki/Phishing
2. https://www.sciencedirect.com/science/article/pii/S1071581918303628
3. https://archive.ics.uci.edu/ml/datasets/Phishing+Websites
4. http://eprints.hud.ac.uk/24330/
5. https://documenter.getpostman.com/view/1034072/S17oxAFE
6. https://www.semrush.com/analytics/backlinks
7. https://toolbox.googleapps.com/apps/messageheader/
8. https://www.iptrackeronline.com/email-header-analysis.php