

**MULTIMEDIA UNIVERSITY OF KENYA**

FACULTY OF COMPUTING & INFORMATION TECHNOLOGY

**PHISHING DETECTION SYSTEM**

BY

**NAME: SIMON NZESI MUTUA**

**REG. No: CIT-227-010/2017**

**SUPERVISOR: DR. ODEO**

November, 2021

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted in partial fulfillment of the requirements of Bachelor of Science in Software Engineering.**

# DECLARATION

I hereby declare that this Project is my own work and has, to the best of my knowledge, not been submitted to any other institution of higher learning

.

Student Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Registration Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: ............................................... Date:.....................................................

This project proposal has been submitted as a partial fulfillment of requirements for the Bachelor of Science in Software Engineering of Multimedia University of Kenya with my approval as the University supervisor.

Supervisor: **DR. ODEO**

Signature: ..................................................... Date: ..................................................

# ACKNOWLEDGEMENTS

My greatest gratitude goes to God for making this research work possible. I would like to acknowledge the Multimedia University of Kenya, and the entire staff for their support throughout this study. I also thank my supervisor Dr. Odeo for his guidance. Special gratitude to my undergraduate colleagues for their support and academic guidance throughout the study.

# ABSTRACT

The Internet has become an indispensable part of our life, However, It also has provided opportunities to anonymously perform malicious activities like Phishing. Phishers try to deceive their victims by social engineering or creating mock up websites to steal information such as account ID, username, password from individuals and organizations. Although many methods have been proposed to detect phishing websites, Phishers have evolved their methods to escape from these detection methods. Mostly, these methods rely on blacklisting and whitelisting techniques giving the attackers a chance to easily bypass the implemented security methods. Also these methods are very specific to certain phishing instances only and hence do not detect zero day attacks.

One of the most successful methods for detecting these malicious activities is Machine Learning. This is because most phishing attacks have some common characteristics which can be identified by machine learning methods. This paper focuses on the use of machine learning methodology in conjunction with a browser extension that runs on the client side (browser) to effectively check all the URLs that the user visits and determine if those URLs are meant for phishing attacks. The browser extension then flags such URLs and warns the user of the possible phishing attack.

# List Of Abbreviations

ANN – Artificial Neural Networks

API – Application Programming Interface

COVID-19 – Coronavirus Disease

DNS -Domain Name System

HR – Human Resources

ICT – Information and Communication Technology

IDN – International Domain Name

IoT –Internet of Things

IP – Internet Protocol

ML – Machine learning

SEA – Social Engineering Attacks

SQL – Structured Query Language

SVM –Support Vector Machines

URL – Uniform Resource Locator

WWW -World Wide Web

XSS – Cross-site Scripting Attacks

**Table of Contents**

[DECLARATION 2](#_Toc88068813)

[ACKNOWLEDGEMENTS 3](#_Toc88068814)

[ABSTRACT 4](#_Toc88068815)

[List Of Abbreviations 5](#_Toc88068816)

[Table of Contents 6](#_Toc88068817)

[List Of Tables 8](#_Toc88068818)

[List Of Figures 9](#_Toc88068819)

[CHAPTER 1: INTRODUCTION 10](#_Toc88068820)

[1.1 Background of Study 10](#_Toc88068821)

[1.2 Problem Statement 11](#_Toc88068822)

[1.2.1 Proposed solution 12](#_Toc88068823)

[1.3 Aim of the study 12](#_Toc88068824)

[1.3.1 Research Objectives 12](#_Toc88068825)

[1.4 Justification of the Study 13](#_Toc88068826)

[1.5 Scope 13](#_Toc88068827)

[1.6 Assumptions 13](#_Toc88068828)

[1.7 Limitations 13](#_Toc88068829)

[CHAPTER 2: LITERATURE REVIEW 14](#_Toc88068830)

[2.1 Introduction 14](#_Toc88068831)

[2.2 Related Systems 14](#_Toc88068832)

[2.2.1 User Education and Training about Phishing 14](#_Toc88068833)

[2.2.2 Server-side software solutions 15](#_Toc88068834)

[2.2.3 Client-side software solutions 17](#_Toc88068835)

[2.3 Limitations/Weaknesses of these systems 19](#_Toc88068836)

[2.4 How the proposed solution handles these weaknesses 20](#_Toc88068837)

[CHAPTER 3: METHODOLOGY 22](#_Toc88068838)

[3.1 Introduction 22](#_Toc88068839)

[3.2 Software Development Methodology 22](#_Toc88068840)

[3.2.1 Evolutionary Model 22](#_Toc88068841)

[3.2.2 Evolutionary Model Steps 22](#_Toc88068842)

[3.3 Project Resources 24](#_Toc88068843)

[CHAPTER 4: SYSTEM ANALYSIS 25](#_Toc88068844)

[4.1 Introduction 25](#_Toc88068845)

[4.2 Detailed Analysis of the current System 25](#_Toc88068846)

[4.2.1 Use case Diagram 25](#_Toc88068847)

[4.2.2 Sequence Diagram 26](#_Toc88068848)

[4.3 System Requirements 26](#_Toc88068849)

[4.3.1 Functional requirements 27](#_Toc88068850)

[4.3.2 Non-Functional requirements 27](#_Toc88068851)

[CHAPTER 5: SYSTEM DESIGN 28](#_Toc88068852)

[5.1 Overview 28](#_Toc88068853)

[5.2 Architectural design 28](#_Toc88068854)

[5.2.1 Architectural design diagram 28](#_Toc88068855)

[5.2.2 Architectural design description 29](#_Toc88068856)

[5.3 User Interface design 29](#_Toc88068857)

[CHAPTER 6: IMPLEMENTATION AND TESTING 31](#_Toc88068858)

[6.1 Overview 31](#_Toc88068859)

[6.2 Development environment 31](#_Toc88068860)

[6.2.1 Dataset 31](#_Toc88068861)

[6.2.2 Machine learning Implementation 33](#_Toc88068862)

[6.3 System components 34](#_Toc88068863)

[6.4 Test Data 36](#_Toc88068864)

[6.5 Test Results 37](#_Toc88068865)

[CHAPTER SEVEN: CONCLUSIONS 40](#_Toc88068866)

[7.1 Achievements and lessons learnt 40](#_Toc88068867)

[7.2 Conclusions 40](#_Toc88068868)

[7.3 Recommendations 41](#_Toc88068869)

[References 42](#_Toc88068870)

[APPENDICES 43](#_Toc88068871)

[Appendix I: Project Schedule 43](#_Toc88068872)

[Appendix II: Project Budget 44](#_Toc88068873)

[Appendix III: Python Program for ML algorithms 45](#_Toc88068874)

# List Of Tables

Table 1: Random forest confusion matrix 37

Table 2: Artificial Neural Network confusion matrix 37

Table 3: SVM confusion matrix 37

# List Of Figures

Figure 1: Evolutionary Model 23

Figure 2:Use case diagram 25

Figure 3: Sequence diagram 26

Figure 4: Architectural design 28

Figure 5: No phishing detected alert window 30

Figure 6: Phishing detected alert window 30

Figure 7: Accuracy of ML classifiers 38

Figure 8: Sensitivity of ML classifiers 39

# CHAPTER 1: INTRODUCTION

## 1.1 Background of Study

The ever-growing and fast expanding reach of the internet is coupled with a rapid spread of information and communication technology (ICT). This advent of new communication technologies has had tremendous impact in the growth and promotion of businesses spanning across many applications including online-banking, e-commerce, and social networking. In fact, in today’s age it is almost mandatory to have an online presence to run a successful venture.

As a result, the importance of the World Wide Web (WWW) has continuously been increasing. Unfortunately, the technological advancements come coupled with new sophisticated techniques to attack and scam users. Such attacks include rogue websites that sell counterfeit goods, financial fraud by tricking users into revealing sensitive information which eventually lead to theft of money or identity, or even installing malware in the user’s system. There are a wide variety of techniques to implement such attacks, such as explicit hacking attempts, drive-by download, social engineering, phishing, man-in-the middle, SQL injections, loss/theft of devices, denial of service, distributed denial of service, and many others.

Most of these attacking techniques are realized through spreading compromised URLs. URL is the abbreviation of Uniform Resource Locator, which is the global address of documents and other resources on the WWW. A URL has two main components : protocol identifier (indicates what protocol to use also known as scheme) and a resource name. The resource name component includes the subdomain, domain name, top level domain, port, path, query, parameters and fragment segments.

Compromised URLs that are used for cyber-attacks are termed as malicious URLs. The most popular type of attack using malicious URLs is the phishing attacks.

Phishing is a social engineering attack which aims at exploiting weaknesses found in system end users. In phishing, an attacker attempts to steal user’s private, sensitive data and tries to manipulate the data by using a malicious fake link which exactly looks like a legitimate link (i.e. the attacker masquerades as a trusted entity/reputable source). When a user clicks on that link, he/she is redirected to the hacker's webpage instead of the legitimate one. User thinking that it’s legitimate provides them the sensitive data such as login credentials, credit card numbers, email addresses and phone numbers and hence user’s data gets compromised.

With the outbreak of the COVID-19 pandemic, a significant rise of phishing attacks has been observed. The global impacts of coronavirus, as well as its implications, such as quarantine measures, a remote workforce, and the length of the pandemic are the core reasons for the increased cases. According to WHO, hackers and cyber scammers are taking advantage of the pandemic by sending fraudulent email and WhatsApp messages that attempt to trick you into clicking on malicious links or opening attachments aimed at carrying out a phishing attack.

For instance, a spam and phishing report by Kaspersky has revealed that they detected 2,023,501 phishing attacks in South Africa, Kenya, Egypt, Nigeria, Rwanda and Ethiopia in the second quarter of 2020. According to the report, South Africa had the biggest number of attacks at 616,666 within three months. This was followed by Kenya with 514, 361 attacks, Egypt at 492,532 attacks, Nigeria at 299,426, Rwanda at 68,931 and Ethiopia at 31,585. The communication to unsuspecting users was disguised as delivery services, postal services, financial services and HR services triggered by embracement of remote workforce across the globe.

## 1.2 Problem Statement

Phishing has been around and it continues to present a substantial risk to businesses as well as individuals and is often cited as a top security concern. The concern is driven by increasingly sophisticated attacks; the move from email to alternative attack vectors, such as social media and messaging; and the simple fact that phishing targets the weakest link in the security chain: people.

The global situation that has come into existence due to the COVID 19 pandemic has changed the equation in the space of Social Engineering attacks. The increase in Work-from-home situations, online education, and entertainment via online platforms has created a sharp uptick in the number of Internet Users worldwide and also consequently increasing the phishing attacks.

Phishing, one of the staples in the SEA arsenal has seen a huge increase, with technology companies such as Google and Microsoft recording trends where the attackers masquerade as officials from organizations working on COVID-19 such as the World Health Organization. As a result, phishing attacks that use social media and messaging to launch and amplify an attack are on the increase.

Often, internet users make typing errors while entering the website URL which is exploited by attackers. Besides these, the attacker may also choose to manipulate the URL by altering the sub-domain names, query lengths, adding redirect requests or making the URL excessively long. Since phishing data is easily available in phishing databases such as Phishtank, once a website is suspected/flagged of being related to phishing, the attacker can easily modify the website URL by altering the sub-domain names to make a new website. Therefore, there is a need for an intelligent method for identifying phishing URLs and reduce phishing attacks. Machine learning techniques can help in classification and prediction of website URLs into phishing and legitimate URLs

### 1.2.1 Proposed solution

The proposed solution is a browser extension framework that works at the client side (browser) of the user accessing web pages from the internet.

This solution uses machine learning to overcome the drawbacks associated with the traditional approaches to phishing detection. The problem of phishing detection is an ideal candidate for the application of machine learning solutions because of the easy availability of sufficient amounts of data on phishing attack patterns.

The basic idea is to use machine learning algorithms on available datasets of phishing

webpages to generate a model which can be used to make classifications in real time if a given web page is a phishing site or a legitimate webpage.

## 1.3 Aim of the study

The main goal of this study is to develop an efficient phishing detection system to allow a user who visits a URL on the Internet determine whether a URL is a phishing or legitimate one by ensuring high accuracy levels while reducing the average computational cost.

### 1.3.1 Research Objectives

1. To develop a phishing detection system that promptly alerts the user in case of a phishing instance.
2. Design a browser extension that is capable to handle real-time scenarios and zero-day attacks.
3. To train and test a machine learning model for feature extraction and classification in real time to cater for new websites that causes zero-day phishing attacks.

## 1.4 Justification of the Study

Most of the anti-phishing solutions in literature claim high accuracy as 98% for phishing detection but most of these measures fail to handle real-time zero-day attacks. There is a huge gap between the high accuracy that has been reported in articles but when it comes to real-time scenario implementation, most of the existing solutions have very low effectiveness. A major reason for low-effectiveness of most of the existing solutions is; the design and ideology behind the solution is influenced by high detection accuracy obtained by training the classifiers with datasets having limited features or spatial correlation.

An effective anti-phishing solution should be characterized by: the detection performance should be evaluated in real time scenarios after considering all the use cases and deployment cases and the evaluation or assessment methodology should be fast enough to provide efficient results in fractions of seconds.

The proposed system sees into it that these requirements are attained by utilizing the advantages of machine learning classifiers in the process of phishing detection.

## 1.5 Scope

The proposed system covers all the URLs that the user visits using their web browser. This means both those that the user opens by clicking from other sources such as email clients or manually typing URL in the browser address bar.

## 1.6 Assumptions

The system does not block a web page after flagging it as a phishing URL. Assumption is hereby made that it is the user’s role to avoid continued browsing to such a website or web page after the system alerts the user of the possible phishing scenario.

## 1.7 Limitations

The system is only fully advantageous to the user when using a web browser in a PC, a laptop or a tablet. This is because of the default behavior of most of the mobile-based web browsers not able to install and run browser extensions.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

Automatically detecting phishing web pages has attracted much attention from security and software providers, financial institutions, to academic researchers. This section analyzes and discusses the several approaches and comprehensive strategies that have been suggested to tackle phishing attacks through each level of attack flow. This section also outlines the weaknesses of these methodologies and how the proposed system handles these issues to bridge the gaps.

## 2.2 Related Systems

The solutions that have been proposed in literature and in industry can be grouped into three main categories:

User education and training

Server-side software solutions

Client-side software solutions

### 2.2.1 User Education and Training about Phishing

Phishing attacks usually target the users who are not much aware of the defense mechanisms for phishing. One of the effective solutions for phishing is thus training and describing to users the fact that not to follow links blindly to any web sites where they have to enter sensitive information such as passwords (Cranor 2007). In one of the 2010 studies, the author has found that training users is useful and help people to identify the fake website provided the users read the phishing material seriously and understand this crime (Sheng, 2010).

Educating users and company employees and warning them about phishing attacks have an impact on preventing phishing attacks. Multiple methods have been proposed for training users. Many researchers concluded that the most impactful approach to help the users to distinguish between phishing and legitimate websites is interactive teaching (Dodge Jr, 2007).

Although user training is an effective method, however, human errors still exist and people are prone to forget their training. It has also been noted that providing anti-phishing material and training users are sometime ineffective since the users are more familiarized of receiving such warnings and thought they already knew how to protect themselves. Training also requires a significant amount of time and it is not much appreciated by non-technical users.

### 2.2.2 Server-side software solutions

Server-side solutions are server-based applications that attempt to mitigate the phishing problem. The idea behind server-side anti-phishing solutions is to protect a user from being a victim of a phishing attack by filtering incoming emails, taking action against fraudulent websites, or applying authentication protocols at the recipient’s mail server. These solutions make use of email-content analysis, notice-and-take-down, or protocol-based authentication methods.

#### 2.2.2.1 Email-content analysis method

The email-content analysis method focuses on examining incoming emails to find specific features of fake emails to prevent such emails from reaching the user’s inbox. To determine these features, a number of known fake emails are analyzed. The model-based machine learning and property-structure based techniques are the examples of the various techniques associated with this method.

Bergholz et al. (2008) proposed a model-based machine learning technique. In this technique a new email’s features are compared to features of known phishing emails. Then a judgment on the new email is made as to whether this email is fake or normal. This technique uses 27 basic features and different advanced features. The basic features can be grouped into five categories: structural, link, element, spam filter-based and word-based. The advanced features are proposed by the authors. They adaptively trained Dynamic Markov Chains and novel latent Class-Topic Models to generate these features . To compare the new email’s features to the proposed features, the technique uses a classifier. Typically, this classifier has two inputs: the values of the phishing emails’ features (the training set of the classifier), and the values of the new emails’ features (the test set of the classifier).

Another technique is proposed by Chandrasekaran et al. (2006). This technique makes use of the structural properties of phishing emails to distinguish between legitimate and fake emails. To achieve their target the authors have identified 25 features. These features can be grouped into two categories: style markers-based (I.e Total number of characters, Total number of unique words, Word count, Total number of function words, Function word frequency distribution, Account, Log, Access, Bank, Credit, Click, Identity, Inconvenience, Information, Limited Minutes, Password, Recently, Risk, Social, Security, Service, Suspended, Total number of words) and structural attributes-based features ( Structure of email subject line, Structure of the greeting provided in the email body). The authors used 100 phishing and 100 legitimate emails as input to the simulated annealing algorithm, to identify the useful features. From the relevance between such features, information gain (IG) has been used to rank these features. Based on the candidate features, the authors used the Support Vector Machine (SVM) classifier to classify phishing emails.

#### 2.2.2.2 Notice-and-take-down method

Another method to combat phishers is to attack their websites before they can start harming any individuals. This can be done by finding these websites’ URLs from reported phishing emails, for example, then try to remove these websites from the Internet. Typically, specialist companies play this role as a service to financial organizations.

#### 2.2.2.3 Authentication protocol method

This method tries to solve the phishing problem by adopting authentication schemas. These schemas can be applied on the email protocol (SMTP), which is designed without security requirements. Using this method, sender’s identity can be examined. This can mitigate phishing risks. A number of techniques, that adopt this method, have been proposed such as senderID by Microsoft and DomainKeys Identified Mail (DKIM) by Yahoo.

Sender ID is used to detect spoofing. Sender ID uses the RECEIVED SMTP header and a query to the DNS records for the sender's domain to determine if the sender's email address is spoofed. Sender ID in Exchange Server is provided by the Sender ID agent, and is basically unchanged from Exchange Server 2010. When the Exchange server receives an inbound message, the Sender ID agent verifies the sender's IP address by querying the DNS records for the sender's domain. This check confirms that the message was received from an authorized IP address for the sender's domain. The IP address of the authorized sending server is referred to as the purported responsible address (PRA).

### 2.2.3 Client-side software solutions

Client-based solutions are designed to work on the Internet users’ machines. That is, using plug-ins or browser helper objects (BHOs – for Microsoft Internet Explorer) which a user can install to monitor visited web pages, and to warn the users if they have entered a fraudulent page. These solutions are different in terms of how to determine if a visited page is fraudulent or not. They can be classified into four groups: blacklist-based or white-list-based, visual-clue-based, webpage-feature-based and information-flow-based solutions.

#### 2.2.3.1 Blacklist-based method

The majority of anti-phishing methods rely on a blacklist, a list of known phishing domains. This method combats the phishing attempts by preventing user from accessing web pages that appear in the blacklist. To build this list, the method requires retrieving recent URLs of phishing webpages from specialist websites such as Anti-phishing Working Group (APWG) or PhishTank, or alternatively may receive these URLs from the users directly. The technique of Microsoft SmartScreen component make use of blacklists method.

Microsoft SmartScreen is integrated with the Internet Explorer and their modern web browser, Microsoft Edge. This tool uses two methods to determine the nature of a page: blacklist checking and heuristics analyses. Basically, when a user visits a site using IE9/Edge, the SmartScreen Filter will compare a page’s contents against heuristics characteristics, which are updated periodically using machine learning techniques developed by Microsoft. If suspicious properties are found, the tool will warn the user to avoid providing any confidential data by causing a yellow shield to appear. However, if the page passes the heuristics test, the tool will check its URL against a frequently updated online blacklist. If the URL is found in the blacklist, the page’s contents will be blocked, and a red shield will appear in the address bar. The user then has the choice whether to proceed or to close the page. The tool also checks downloaded files against the same blacklist, and the later processes will be applied. SmartScreen Filter provides its user with a reporting feature to notify Microsoft about new fraudulent URLs. In addition, to decrease the false positive detection rate, this tool depends only on verified unsafe URLs provided by reviewers at Microsoft or by employees from third parties.

#### 2.2.3.2 Visual-clue-based method

Visual-clue-based method applies the idea of using images as a base for the solution to combating phishing attacks. This method relies on the fact that phishing attackers try to lure users by imitating visual features of target websites. This method tends to use images as authentication evidences that the server should present. Visible Watermarking is an example of a visual-clue-based techniques.

Visible watermarking (ViWiD) is an integrity check technique in which the user needs to verify a watermark within the company webpage’s logo to authenticate this webpage (M. Topkara, 2005). The company’s logo can appear to the user in two ways: after the user login into his or her account, or by using a cookie. The last choice is preferred since the user need not to enter his or her confidential data on the login webpage to avoid revealing this data on a forged webpage. The user can trust the server’s webpage since its logo includes the shared secret. The process of adding the watermark to the company’s logo is done on the company web server, and the user need not install any tool or store any data on his or her local machine.

#### 2.2.3.3 Webpage-feature-based method

Another method depends on analyzing the web-page's contents to find fraud symptoms, and then warning the user of a potential phishing attack. A number of techniques that adopt this method have been proposed, for example SpoofGuard.

SpoofGuard is used in mitigating simple phishing attacks (Chou, 2004). The plug-in monitors a user’s Internet activity, computes a spoof index, and warns the user if the index exceeds a level selected by the user. SpoofGuard uses domain name, URL, link, and image checks to evaluate the likelihood that a given page is part of a spoof attack. For ex-ample, a page with a suspicious URL such [*https://asetrade*](https://asetrade/)[*maintenance.suspicious.orgorwww.etrade.com@129.170.213.101maintainance.asp*](mailto:maintenance.suspicious.orgorwww.etrade.com@129.170.213.101maintainance.asp)and an E\*Trade logo will have a higher spoof index than a page with neither of these characteristics.

Therefore, some of these evaluations are done after downloading the webpage: URL, link, image and domain checks. In addition, some evaluations are conducted when the user interacts with such a page: password, outgoing password, referring page, outgoing post data checks.

#### 2.2.3.4 Information-flow-based method

Information-flow-based method tries to protect users from being victims of phish attacks by tracking their sensitive information to make sure that they provide this information on trusted websites. A user will be warned, if he/she is about giving away her confidential data on fake websites. One technique that follows this method is AntiPhish (Kirda, 2005). This technique detects phishing by examining the current web-page's domain when a user starts to enter sensitive data.

The AntiPhish technique’s main purpose is to protect users’ confidential data. This can be done by monitoring where the users’ confidential data is being entered and informing the user in the case of a phishing attack. Typically, when a user enters confidential data in a web page’s form for the first time, he/she may ask AntiPhish to capture this data and stores it in an encrypted form. AntiPhish also stores a web page’s domain to be mapped with the user data. AntiPhish uses a domain rather than a web page’ address because some websites are hosted in more than one server. To monitor the users’ confidential data, AntiPhish examines text field elements of any form in a web page and interrupts any user event. If the user interacts with a text element, AntiPhish will compare the element value against a list of previous stored user’s confidential data. If it finds a match, domains comparison will start. If there is no match, AntiPhish will consider the current webpage as phishing. AntiPhish runs same test if the user generates events on test elements: press a key, load new page, click or focus. JavaScript gives an attacker the ability of accessing form’s text elements before a user submits inputs. To combat this problem, AntiPhish deactivates JavaScript if the focus is on a text element and reactivates it when the focus is lost.

## 2.3 Limitations/Weaknesses of these systems

#### 2.3.1 User Education

People are prone to forget their training.

Training also requires a significant amount of time and it is not much appreciated by non-technical users.

#### 2.3.2 Server-side software solutions

The email-content analysis method may not identify some browser vulnerabilities-based attacks , such as International Domain Name (IDN) spoofing and pop-up hijacking attacks. This is because the proposed technique focuses only on email-based attacks.

The notice-and-take down methods require Server Administrator interaction to remove phishing pages thus affecting the performance of the solution.

The authentication protocol methods, sometimes the forwarding services need to modify a message’s content. In many instances this modification will result in authentication fail.

#### 2.3.3 Client-side software solutions

As with any blacklist-based solution, users are still exposed to new phishing attacks. That is, the URLs of newly established phishing sites may not yet be included in the blacklist.

The Visual-clue-based technique requires the user to have some knowledge of phishing attacks and how to identify spoofed pages in order to distinguish between an authentic and a spoof webpage. As a result of the leak knowledge, more than 20% of users ignore web-page’s visual clues and even professional users may be victims of visual-based attacks.

For the Visible-watermarking, the users have to be trained to expect what information should appear in the company’s logo in order to distinguish between real and fake web pages.

SpoffGuard can be fooled using sophisticated hybrid phishing attacks such as Cross-Site Scripting (XSS) because it is developed to address simple phishing attacks. Also, it it platform dependent as it only works with Microsoft Internet Explorer.

In Information-flow-based method, the user needs to inform AntiPhish to capture their confidential data. I.e the user participates in the verification process.

## 2.4 How the proposed solution handles these weaknesses

Various anti-phishing solutions proposed by different authors have been given in the previous section. However, no single solution is a “full proof” solution for combating phishing attack. The Limitations of the existing anti-phishing solutions outlined above emphasize the need for innovative solutions.

The proposed anti-phishing solution will be implemented at the client-side in the form of a browser extension and should be capable to handle real-time scenarios and zero day attacks. The proposed approach works efficiently for any phishing link carrier mode as the execution on clicking on any link or manually entering URL in the browser automatically launches the evaluation process to determine the legitimacy of the specified URL. This ensures that the user needs not to manually be involved in verifying if a URL is genuine or not as long as the browser extension is turned on.

Since client-side solutions in the form of browser extensions are easy to install and use, this ensures that the computational costs are kept at minimum thus increasing the efficiency of the proposed solution.

The proposed solution makes use of machine learning which makes it easy to detect zero-day phishing attacks. This is because most phishing attacks have some common characteristics which can be identified by machine learning methods. As a result, the system can make fast decisions in future of determining the legitimacy of a URL based on the previously identified instances of such URLs.

# CHAPTER 3: METHODOLOGY

## 3.1 Introduction

Software development methodology is the process of dividing software development work into smaller, parallel or sequential steps or sub-processes to improve design, product management and project management.

## 3.2 Software Development Methodology

This project will focus on the use of the evolutionary model of software development.

### 3.2.1 Evolutionary Model

Evolutionary model suggests breaking down of work into smaller chunks, prioritizing them and then delivering those chunks to the customer one by one. The number of chunks is huge and is the number of deliveries made to the customer. The main advantage is that the customer’s confidence increases as he constantly gets quantifiable goods or services from the beginning of the project to verify and validate his requirements. The model allows for changing requirements as well as all work in broken down into maintainable work chunks.

The choice of this methodology comes handy due to the time factor involved; the project is estimated to run for the next ten weeks which is a major constraint in the development.

Due to the reason stated, the use of iterative model will allow for instant modification upon and during development and in addition the application can be altered easily. Evolutionary model will allow growth of this project in a steady way.

### 3.2.2 Evolutionary Model Steps

The following are the steps to be taken to complete the development of the project:

1. Drafting of the Project proposal
2. Analysis of requirements and software specification
3. Design of different Module and interface
4. Implementation through Coding
5. Testing the application
6. Verification and validation
7. Documentation

The following section illustrates and discusses these steps in details:

Drafting Project Proposal

Requirements and software specifications

Analysis and Design

Implementation

Testing

Verification and Validation

Documentation

YES

NO

Figure 1: Evolutionary Model

*Requirements and software specification*

This is the first stage of development. It will involve researching and analyzing the requirements for the project to develop a list of requirements for the project.

*Design of different modules and interface*

This is the second phase of the development. After the requirement phase is complete, the requirements are analyzed and an initial design is created. This will be a simple design.

*Implementation*

This stage will include coding activities and use of python environment for code generation, guided by the initial design created in the second phase. Every specification will be applied into the program structure. If there are any errors with the design, the project will return to the design stage and a reviewed design will be produced.

*Testing*

This stage will check for the working of the functionalities implemented in the coding phase to test whether or not they work as expected and correct any errors arising from the coding activity itself.

*Verification and validation*

This stage will examine whether the prototype developed meets all the requirements of the project. If it does not meet the requirements, the process will have to take an entire iteration if the prototype meets all the requirements of the project, then the process moves to the next stage. This will also ensure the project is in due course with the referenced design from the earlier stages.

*Deployment*

The application will then be rolled out to the students in multimedia for their trial once the development time is reached. The students will then be able to give feedback and support in future development of the project.

Hopefully I will be able to integrate a simple feedback module so as to hear what people think of the project

## 3.3 Project Resources

*Hardware Requirements*

A computer/laptop

*Software Requirements*

Python 3.8

Visual Studio Code

Web Browser

JavaScript Framework

UCI Machine learning repository phishing websites dataset

# CHAPTER 4: SYSTEM ANALYSIS

## 4.1 Introduction

This chapter offers a general overview of the model based on its design and application. This section also outlines the system requirements in terms of; data, functional and non-functional requirements.

## 4.2 Detailed Analysis of the current System

This section distinctively showcases the various pictorial and graphical representations and diagrams that are used in the design stage such as use case diagrams, sequence diagrams, activity diagram, data flow diagrams, class diagram and the entity relation diagram. These diagrams help to further define and understand the model.

### 4.2.1 Use case Diagram

Careful modeling of system is crucial to obtain the correct and most efficient model architecture. A use case diagram captures the dynamic view of a system by defining the actions and behaviors of a system as it interacts with some external parties (users).

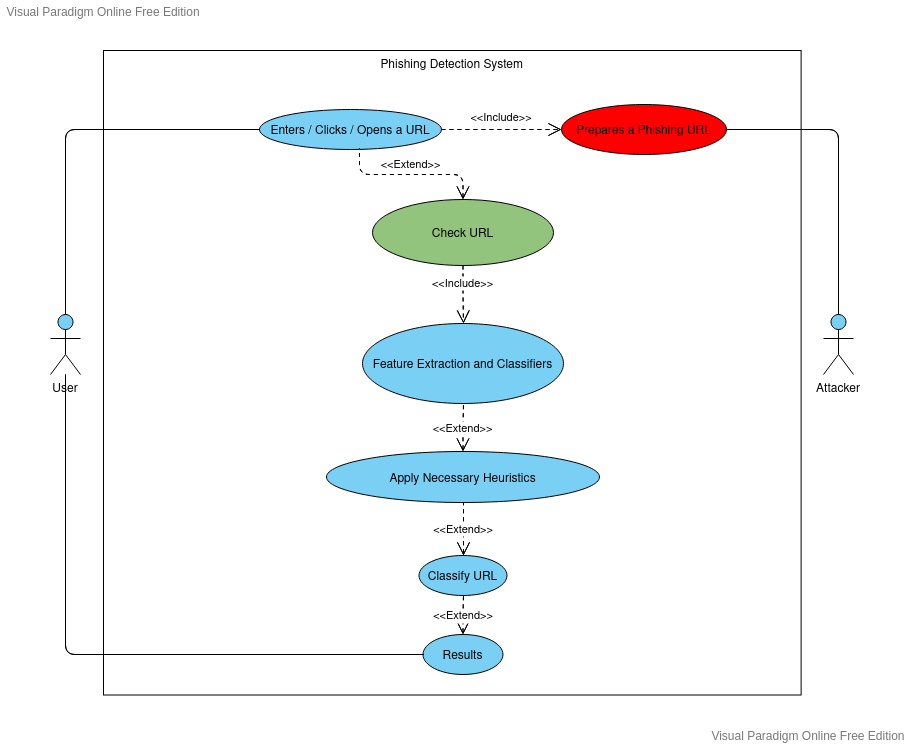


Figure 2:Use case diagram

### 4.2.2 Sequence Diagram

Sequence diagrams are commonly used to represent the flow of actions, events and messages across a software system as well as to design, document and even validate a system’s architecture.

The sequence diagram below highlights the flow of actions, events and messages across the phishing detection system.

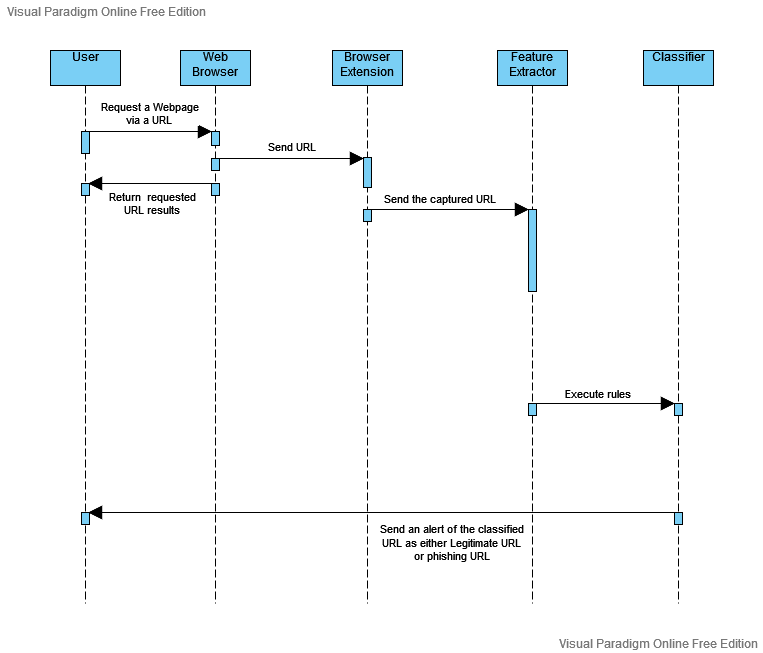


Figure 3: Sequence diagram

## 4.3 System Requirements

Phishing takes on multiple forms depending on the vulnerabilities exploited by the phishers. As highlighted in chapter 2 of this report, various exploits are used to target and steal personal information from users. This component forms the basis of the requirement analysis which then defines the appropriate resources needed to meet the objectives set by this study.

Under this section, both the functional and non-functional requirements of the phishing detection model are discussed.

### 4.3.1 Functional requirements

These are requirements that capture specific behavior of the system under development. These define various things like data processing and manipulation, system calculations and interaction with application and other functionality that show how user requirements are satisfied.

The functional requirements of this system are:

1. The model should have direct access to user’s requested URL immediately after the user enters or clicks or opens the URL on a web browser.
2. URL shortening threat detection function.
3. Extraction of URL features and details.
4. Flag identified phishing URL details and features and then alert the user.
5. Provide feedback of system performance e.g presence of false positives.

### 4.3.2 Non-Functional requirements

1. A high usability based on its ease of use.
2. The model should be reliable and provide accurate results.
3. The model must be always available to the end user.
4. The model must maintain the security of the user (their emails etc.).
5. A high interoperability.
6. Recoverability and maintainability.

# CHAPTER 5: SYSTEM DESIGN

## 5.1 Overview

In the evolutionary software development life cycle, the implementation stage is the equivalent of the system design process. It is the phase where computer and paper-based models of how the system will look like and work are created. It explains all of the systems components and functions, relationships between components and what the interface looks like. Anyone should have a basic understanding of the system by just studying the diagram and models presented under system design section.

### 5.2 Architectural design

### 5.2.1 Architectural design diagram

The basic architecture of the proposed system is shown in the figure below.

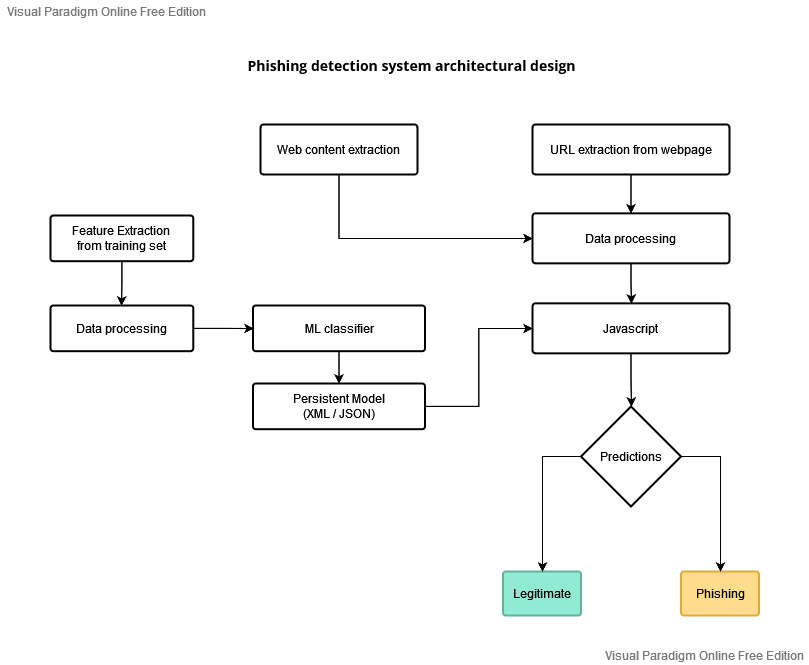


Figure 4: Architectural design

### 5.2.2 Architectural design description

*Overview*

The proposed approach aims at building a browser extension powered by state-of-the-art machine learning technique for phishing detection. Furthermore, given the flexibility of margin and reduced computational complexity offered by SVM, for classification problem statements, the implementation employs SVM trained persistent model to identify the malicious sites.

The extension is packaged to support Chrome browser in specific, solely by the virtue of its popularity. Additionally, extensions exhibit minimal web-dependence, as it collates multiple files into single file for user to download, as one-time activity.

*How the architecture works*

The solution deals with training the model with available data-set, using SVM discriminative classifier, followed by passing the persistent model to the extension, which further predicts the authenticity of the user accessed websites and provides alerts to notify the legitimacy of the browsed URL on every page load. The solution integrates Python-based training stage implementation with JavaScript-based testing module. The training component has been designed using Python, so as to make optimal utilization of the available complex numeric computation libraries. Moreover, given the fact that the testing stage is centric to web-content and feature extraction, and has minimal heavy computation activities associated; the solution does face client-end computation performance lag concerns.

## 5.3 User Interface design

The proposed system consists of an alert window that notifies the user whether the URL is a legitimate one or a phishing URL while they are browsing.

The following screenshots shows how these pop-up windows are designed as viewed by the client.

*Pop window when user visits a legitimate URL*

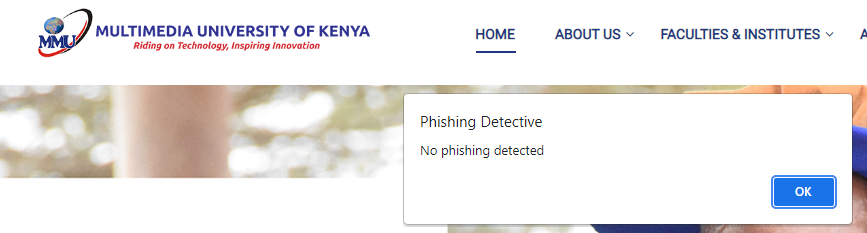
**

Figure : No phishing detected alert window

*Pop window that warns user of a phishing webpage*

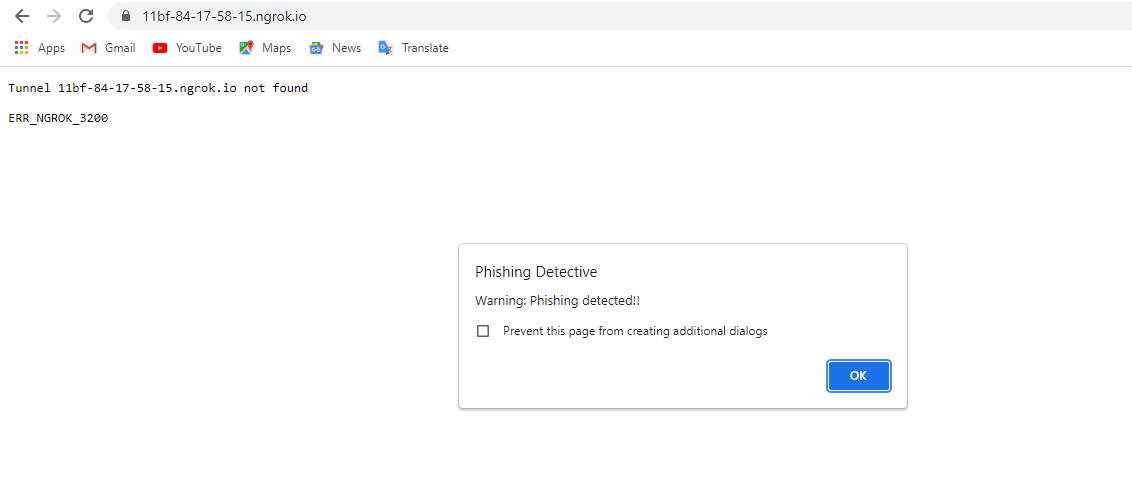


Figure : Phishing detected alert window

# CHAPTER 6: IMPLEMENTATION AND TESTING

## 6.1 Overview

This chapter discusses about the testing of the solution and implementation methodologies. This constitutes the dataset used, the features of URL extracted, the algorithm employed to train the model and finally the tests carried to ensure that the system meets the specified requirements

## 6.2 Development environment

### 6.2.1 Dataset

The training dataset for our project is taken from the "Phishing Websites Data Set" of the UCI Machine learning repository. The dataset consists of 11,055 entries with 6157 phishing instances and 4898 legitimate instances.

Each instance consists of 30 features comprising of various attributes typically associated with phishing or suspicious web pages such as presence of IP address in the URL domain or presence of JavaScript code to modify the web browser address bar information. Each feature is associated with a rule. If the rule is satisfied, then it is taken as an indicator of phishing and legitimate otherwise.

The dataset has been normalized to contain only discrete values. Each feature of each instance will contain ’1’ if the rule associated with that feature is satisfied, ’0’ if partially satisfied and ’-1’ if unsatisfied.

The features represented by the training dataset can be classified into four categories;

1. Address Bar based features
2. Abnormal based features
3. HTML and JavaScript based features
4. Domain based features
5. **Address bar based features**

*1.1 Using IP address*: If the domain of the URL of the suspected web page contains IP address, then we take it as a phishing page. eg: http:125.98.3.123fake.html or http:x58.0xCC.0xCA.0x622paypal.caindex.html

*1.2 Long URL to hide suspicious part:* It has been a common observance that phishing web pages usually have long URLs that attempt to hide malicious URL fragments from the user. We take the assumption that a web page with a long URL is necessarily a phishing or suspicious site. In the event the assertion fails, i.e, for a legitimate web page with valid long URLs, the absence of other phishing attributes on the web page will balance the wrong assumption and correctly classify a legitimate web page as non-phishing.

*1.3 Use of URL shortening services:* A shortened URL hides the real URL behind a redirection hop. A web page that uses a URL shortening service such as TinyURL is highly suspicious and is likely to be a phishing attempt. Therefore, we set the rule that if the URL has been shortened using a URL shortening service then it is a phishing page and legitimate otherwise.

*1.4 Use of "@" symbol:* Needs verification The "@" symbol is a reserved keyword according to Web standards. So the presence of "@" in a URL is suspicious and the web page is taken as phishing and legitimate otherwise.

*1.5 Redirection with "":* The presence of "//" in the URL path indicates the page will be redirected to another page. If the position of "//" in the URL is greater than seven then it is

a phishing site and legitimate otherwise.

*1.6 Adding prefix or suffix separated by "-" to the domain:* Phishers tend to add a prefix or suffix to the domain with "-" to give the resemblance of a genuine site. Eg: www.a-paypal.com

*1.7 Sub domains and multi sub domains:* If a URL has more than three dots in the domain part then it is considered as a phishing site and legitimate otherwise.

1. **Abnormal based features**

*2.1 Request URL:* A legitimate site usually has external page objects such as images, animations, files, etc. be accessed by a request URL which shares the same domain as the web page URL. We classify sites which fail this rule as phishing.

*2.2 URL portion of anchor tag:* We check if the domain in the URL portion of all anchor tags match the main URL of the page and if the anchor tag has only URL fragments or JavaScript functions.

*2.3 Links in <meta>, <script> and <link> tags:* We check if the domain of the links in the <meta>, <script> and <link> tags matches the domain in the mail URL.

*2.4 Server Form Handler (SFH):* When a form is submitted, some valid action must be taken. So if the action handler of a form is empty or "about:blank" or if the domain of the action URL is different from the domain of the main URL, then it is taken as a phishing site.

*2.5 Submitting Information to Email:* If the webpage contains a "mailto:" function then it is taken as a phishing site and legitimate otherwise.

1. **HTML and Javascript based features**

*3.1 Status bar customization:* Phishers can modify the status bar using JavaScript to show a legitimate URL. By analyzing the "onMouseOver" events in the web page we can determine if such a modification has occurred.

*3.2 Disabling right click option:* Phishers can disable the right click option to prevent the user from checking the source code of the page. This is verified by analyzing the source code.

*3.3 Using pop-up window:* Legitimate sites rarely ask for user info on a pop-up window, whereas phishing sites generally use pop-up windows to get user info.

*3.4 Iframe redirection:* Phishers also use Iframe tags with invisible borders to get user info and redirect to the original site. We analyze the source code to check if Iframe tags are used.

### 6.2.2 Machine learning Implementation

This study has trained and tested supervised machine learning algorithms on the training dataset. The following algorithms were chosen based on their performance on classification problems. The dataset was split into training and test set in the ratio 7:3. The results of the experiment are given in the test results section.

The three classifiers used in this study are discussed below:

1. **Random Forests**

Random forests are classifiers that combine many tree predictors, where each tree depends on the values of a random vector sampled independently.

Trees are always grown and never pruned compared to other tree algorithms. Random forests can handle large number of variables in a data set. Also, during the forest building

process they generate an internal unbiased estimate of the generalization error. In addition, they can estimate missing data well.

A major drawback of random forests is the lack of reproducibility, as the process of building the forest is random Further, interpreting the final model and subsequent results is

difficult, as it contains many independent decisions trees.

1. **Artificial Neural Networks**

A neural network is structured as a set of interconnected identical units (neurons). The interconnections are used to send signals from one neuron to the other. In addition, the

interconnections have weights to enhance the delivery among neurons. The neurons are not powerful by themselves, however, when connected to others they can perform complex computations. Weights on the interconnections are updated when the network is trained, hence significant interconnection play more role during the testing phase.

The commonly used function in neural network research is the sigmoid function. Although

competitive in learning ability, the fitting of neural network models requires some experience, since multiple local minima are standard and delicate regularization is required.

1. **Support Vector Machines**

Support Vector Machine (SVM) is a supervised machine learning discriminative model, which conforms to the principle of drawing separating hyper-plane with maximum safety space, called margin, to minimize the risk of flawed predictions.

These three algorithms are implemented in a python program each on its own function and then trained using one dataset for comparison purposes.

## 6.3 System components

The system developed broadly comprises of two main components:

1. the machine learning algorithms component
2. the browser extension component

**Machine learning algorithm details**

The solution deals with training the model with available data-set, using SVM discriminative classifier, followed by passing the persistent model to the extension, which further predicts the authenticity of the user accessed websites and provides alerts to notify the legitimacy of the browsed URL on every page load. The solution integrates Python-based training stage implementation with JavaScript-based testing module. The training component has been designed using Python, so as to make optimal utilization of the available complex numeric computation libraries. Moreover, given the fact that the testing stage is centric to web-content and feature extraction, and has minimal heavy computation activities associated; the solution does face client-end computation performance lag concerns.

**Browser extension**

The Chrome extension complies to the Google norms and, primarily, consists of three main files: *manifest.json, content.js, background.js.*

The manifest file provides all the meta data information about the extension to Chrome browser. Additionally, it also specifies all the files and other resources associated to the extension.

The content.js file loads on every page in the Chrome browser, post the extension deployment. However, it is an unprivileged module, which has direct access only to the DOM elements and needs supporting files to interact to external APIs and browser user interface manipulation.

The supplementary file background.js aids the content script with these interactions, which is termed as message passing.

Multiple functions have been implemented in the content.js script for web-content and URL feature extraction. Below are the details used to identify phishing portals: *isIPInURL():* Identify presence of IP address in the URL

*isLongURL():* Validate if length of the URL is beyond 75 characters

*isTinyURL():* Identify URLs smaller than 20 charaters

*isAlphaNumericURL():* Check for alphanumeric ’@’ in URL

*isRedirectingURL():* Verify if ’//’ existing within the URL more than once

*isHypenURL():* Check for presence of ’-’ adjacent to domain name in URL

*isMultiDomainURL():* Domain name should be confined to top-level domain, country-code and second-level domain.

*isFaviconDomainUnidentical():* Verify if links on given web-page are loaded from other domains

*isIllegalHttpsURL():* Identify presence of multiple ’https’ in the URL string

*isImgFromDifferentDomain():* Validate if images on given web-page are loaded from other domains

*isAnchorFromDifferentDomain():* Detect if links on given web-page are loaded from other domains

*isScLnkFromDifferentDomain():* Identify if scripts on given web-page are loaded from other domains

*isFormActionInvalid():* Detect invalid/blank form submissions

*isMailToAvailable():* Check for anchor tag incorporating mailto

*isStatusBarTampered():* Validate if onmouseover manipulates the status bar display

*isIframePresent():* Identify sites, which exhibit iframes in the DOM

The extracted features are further passed through the SVM model to identify hostile web URLs.

## 6.4 Test Data

The dataset used in this study has a total of 11055 URLs. This dataset was then split into training and test set the ratio 7:3.

Therefore, a total number of 3317 test samples were used for testing purposes. This project compares the performance of all the classifiers described in section 6.2.2 on the phishing dataset. These algorithms are evaluated on the 3317 test samples using various performance metrics and the test results section below contains the tabulated results with their graphs.

## 6.5 Test Results

**Random forest confusion matrix**

|  |  |  |
| --- | --- | --- |
|  | Predicted phishing URLs | Predicted legitimate URLs |
| Ground truth Phishing URLs | 1249 | 162 |
| Ground Truth Legitimate URLs | 182 | 1680 |

Table 1: Random forest confusion matrix

Table 1 shows the confusion matrix for Random forests. With 1249 true positives, 182 false positives, 162 false negatives and 1680 true negatives.

**Artificial neural network confusion matrix**

|  |  |  |
| --- | --- | --- |
|  | Predicted phishing URLs | Predicted legitimate URLs |
| Ground truth Phishing URLs | 1205 | 250 |
| Ground Truth Legitimate URLs | 170 | 1692 |

Table 2: Artificial Neural Network confusion matrix

Table 2 shows the confusion matrix for Artificial neural network. With 1205 true positives, 170 false positives, 250 false negatives and 1692 true negatives

**SVM confusion matrix**

|  |  |  |
| --- | --- | --- |
|  | Predicted phishing URLs | Predicted legitimate URLs |
| Ground truth Phishing URLs | 1293 | 206 |
| Ground Truth Legitimate URLs | 131 | 1731 |

Table 3: SVM confusion matrix

Table 3 shows the confusion matrix for Support vector machine. With 1293 true positives, 206 false positives, 131 false negatives and 1731 true negatives.

**Performance matrix of classifiers**

|  |  |  |
| --- | --- | --- |
|  | Accuracy (%) | Sensitivity (%) |
| Artificial Neural Network | 87.34 | 83 |
| Random Forest | 89.63 | 86 |
| SVM | 89.84 | 89 |

The performance data above is then plotted using bar graphs as follows:

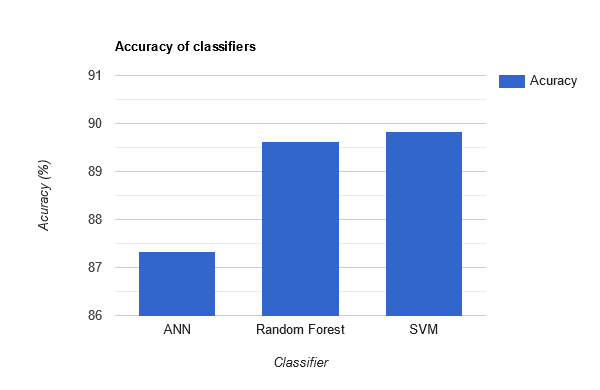


Figure 7: Accuracy of ML classifiers

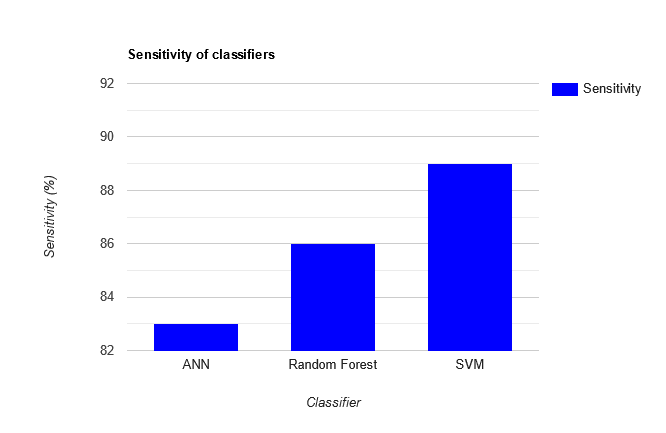
The Figure 3 shows the accuracy of each classifier evaluated on 3317 test samples. From the above figure, it can be seen that SVM outperforms all the other algorithms based on accuracy in detection of Phishing URL. 

Figure 8: Sensitivity of ML classifiers

Figure 4 shows the sensitivity of each classifier. Here sensitivity refers to the classifier’s ability to correctly detect phishing URLs. It can be seen that SVM has the highest sensitivity among all the other classifiers.

However, in phishing detection, false positives and false negatives are given more consideration when studying the performance (predictive accuracy) of a classifier. That is  
because false positives are more expensive than false negatives in the real world. Since we do not want to allow users to access the phishing URLs, false positives are considered to  
be important while deciding the best classifier. It is found out that SVM has the least False positive rate among the three. Hence, SVM works best in classifying the phishing URL from the legitimate URLs

# CHAPTER SEVEN: CONCLUSIONS

## 7.1 Achievements and lessons learnt

With such a project there are so many lessons you learn. The lessons that I have learnt include:

* As I was carrying out this project, I got to interact more with developers from my class as well as developers from other courses and learn more than the things I actually needed which has opened my eyes to the possibilities that exist and also has made me appreciate more about security.
* The web extension was built using the Google chrome extension API. Now I have come to appreciate the sheer simplicity of widely used free and open-source chrome extension API
* When building the machine learning model, I was able to gather skills of handling datasets and separating the same for training and test sets. This contributed to improving my python programming skills as well.

Factoring the lessons learnt from this research work and reported results; the research was able to provide statistically reliable and applicable data sets and results.

In the future after the completion of this project, I want to further explore the world of machine learning together with artificial intelligence and how these two are integrated into IoT.

I am proud that all of the hard work has given me a working end product

## 7.2 Conclusions

To summarize, the study has shown how phishing is a huge threat to the security and safety of the web and how phishing detection is an important problem domain. This paper has also reviewed some of the traditional approaches to phishing detection; broadly categorized into blacklist and heuristic evaluation methods, and their drawbacks.

This paper has tested three machine learning algorithms on the ‘Phishing Websites Dataset’ from the UCI Machine Learning Repository and reviewed their results. Based on the performance of these algorithms, the best was then selected, which is SVM, and built  
a Chrome extension for detecting phishing web pages. The extension allows easy deployment of our phishing detection model to end users.

Based on the research objectives and research findings, the following conclusions are drawn:

* The phishing detection system guarantees security of the end user by sending a pop up alert window to notify the user when a phishing instance is detected while they are visiting various web pages.
* The system is capable of handling future phishing attack methods which have not been updated on phishing data sites such as the Phishtank API. This is because the system analyses the URL features and patterns commonly used by phishers and then learns from such URLs to detect new attack vectors.
* The training process of the model was efficiently done because the system is capable of detecting phishing URLs in real time with little cases of false positives.

However, the second objective may not have been achieved fully. This is because the developed browser extension can only run on Google chrome and chrome-based browsers only. The extension is not compatible with Mozilla Firefox and other browsers which are not chrome-based.

## 7.3 Recommendations

Not a single project is ever considered as complete forever because our mind is always thinking something new and our necessities also are growing day by day. We always want something more than what we have.

I therefore make the following recommendations to the system for future enhancements:

* Conversion of the browser extension to browser independent and cross-platform plugin.
* Build the phishing detection system as a scalable web service which will incorporate online reporting by experienced IT users so that new phishing attack patterns can easily be learned and improve the accuracy of the models with better feature extraction by taking into account every new phishing pattern introduced by phishers.

# References

Bergholz, A., Chang, J. H., Paass, G., Reichartz, F., & Strobel, S. (2008, August). *Improved Phishing Detection using Model-Based Features*. In *CEAS*.

Chandrasekaran, M., Narayanan, K., & Upadhyaya, S. (2006, June). *Phishing email detection based on structural properties.* In *NYS cyber security conference* (Vol. 3).

Dodge Jr, R. C., Carver, C., & Ferguson, A. J. (2007). Phishing for user security awareness. *computers & security*, *26*(1), 73-80.

Kirda, E., & Kruegel, C. (2005, July). Protecting users against phishing attacks with antiphish. In *29th Annual International Computer Software and Applications Conference (COMPSAC'05)* (Vol. 1, pp. 517-524). IEEE.

Cranor, L. F. (2007, May). Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web* (pp. 639-648).

P. Kumaraguru, J. Cranshaw, A. Acquisti, L. Cranor, J. Hong, M. A.  
Blair, and T. Pham, “*School of phish: a real-world evaluation of anti-  
phishing training*,” in Proceedings of the *5th Symposium on Usable  
Privacy and Security*, pp. 1–12, 2009.

N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, “*Client-side defense against web-based identity theft*,” in NDSS. The Internet Society, 2004.

Topkara, M., Kamra, A., Atallah, M. J., & Nita-Rotaru, C. (2005, September). Viwid: Visible watermarking based defense against phishing. In *International Workshop on Digital Watermarking* (pp. 470-483). Springer, Berlin, Heidelberg.

V. B. et al, “study on phishing attacks,” *International Journal of Computer Applications*, 2018.

# APPENDICES

## Appendix I: Project Schedule

This project will use an evolutionary prototyping model to realize the objectives. The estimated development time of the system is ten weeks. The duration of the activities of the model is shown on the gantt chart below

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PHASES** | **ACTIVITY** | **WEEKS** | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| PHASE ONE | Requirement Specification |  |  |  |  |  |  |  |  |  |  |
| Analysis and design |  |  |  |  |  |  |  |  |  |  |
| PHASE TWO | Implementation |  |  |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |  |  |
| Evaluation |  |  |  |  |  |  |  |  |  |  |
| PHASE 3 | Documentation |  |  |  |  |  |  |  |  |  |  |
| System review |  |  |  |  |  |  |  |  |  |  |

## Appendix II: Project Budget

|  |  |
| --- | --- |
| **Resource** | **Cost** |
| Computer/Laptop | Available |
| Python 3.8 | Free |
| Visual Studio Code | Free |
| Internet Connection | 1000 |
| Google Chrome | Free |
| Printing charges | 1000 |
| **Total Cost** | **2000** |

## Appendix III: Python Program for ML algorithms

import time

def calculate\_metrics(y\_test,Y\_predicted):

    from sklearn import metrics

    from sklearn.metrics import classification\_report,confusion\_matrix

    accuracy = metrics.accuracy\_score(y\_test,Y\_predicted)

    print "accuracy = "+str(round(accuracy \* 100,2))+"%"

    confusion\_mat = confusion\_matrix(y\_test,Y\_predicted)

    print confusion\_mat

    print confusion\_mat.shape

    print "TP\tFP\tFN\tTN\tSensitivity\tSpecificity"

    for i in range(confusion\_mat.shape[0]):

        # i means which class to choose to do one-vs-the-rest calculation

        # rows are actual obs whereas columns are predictions

        TP = round(float(confusion\_mat[i,i]),2)  # correctly labeled as i

        FP = round(float(confusion\_mat[:,i].sum()),2) - TP  # incorrectly labeled as i

        FN = round(float(confusion\_mat[i,:].sum()),2) - TP  # incorrectly labeled as non-i

        TN = round(float(confusion\_mat.sum().sum()),2) - TP - FP - FN

        print str(TP)+"\t"+str(FP)+"\t"+str(FN)+"\t"+str(TN),

        sensitivity = round(TP / (TP + FN),2)

        specificity = round(TN / (TN + FP),2)

        print "\t"+str(sensitivity)+"\t\t"+str(specificity)+"\t\t"

    f\_score = metrics.f1\_score(y\_test,Y\_predicted)

    print f\_score

def neural\_network(dataset,class\_labels,test\_size):

    import numpy as np

    import pandas as pd

    from sklearn.cross\_validation import train\_test\_split

    from sklearn.neural\_network import MLPClassifier

    X = pd.read\_csv(dataset)

    Y = pd.read\_csv(class\_labels)

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size= tes t\_size, random\_state=42)

    model = MLPClassifier(hidden\_layer\_sizes=(100), activation='logistic',random\_state = 42)

    model.fit(X\_train,y\_train)

    Y\_predicted = model.predict(X\_test)

    return y\_test,Y\_predicted

def random\_forests(dataset,class\_labels,test\_size):

    import numpy as np

    import pandas as pd

    from sklearn.cross\_validation import train\_test\_split

    from sklearn.ensemble import RandomForestClassifier

    from sklearn import metrics

    X = pd.read\_csv(dataset)

    Y = pd.read\_csv(class\_labels)

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size= test\_size, random\_state=42)

    model = RandomForestClassifier(n\_estimators = 5, criterion = 'entropy',random\_state = 42)

    model.fit(X\_train,y\_train)

    Y\_predicted = model.predict(X\_test)

    return y\_test,Y\_predicted

def support\_vector\_machines(dataset,class\_labels,test\_size):

    import numpy as np

    from sklearn import svm

    import pandas as pd

    from sklearn.cross\_validation import train\_test\_split

    X = pd.read\_csv(dataset)

    Y = pd.read\_csv(class\_labels)

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size= test\_size, random\_state=42)

    # 'rbf' value is the gaussian kernel,

    #'C' is the coefficient used for regularization during training

    model = svm.SVC(kernel='rbf',C=2.0)

    model.fit(X\_train,y\_train)

    Y\_predicted = model.predict(X\_test)

    return y\_test,Y\_predicted

def main():

    dataset = "Dataset.csv"

    class\_labels = "Target\_Labels.csv"

    test\_size = 0.3

    print "\nrunning neural networks..."

    start\_time = time.time()

    y\_test,Y\_predicted = neural\_network(dataset,class\_labels,test\_size)

    calculate\_metrics(y\_test,Y\_predicted)

    end\_time = time.time()

    print "runtime = "+str(end\_time - start\_time)+" seconds"

    print "\nrunning random forests..."

    start\_time = time.time()

    y\_test,Y\_predicted = random\_forests(dataset,class\_labels,test\_size)

    calculate\_metrics(y\_test,Y\_predicted)

    end\_time = time.time()

    print "runtime = "+str(end\_time - start\_time)+" seconds"

    print "\nrunning support vector machines..."

    start\_time = time.time()

    y\_test,Y\_predicted = support\_vector\_machines(dataset,class\_labels,test\_size)

    calculate\_metrics(y\_test,Y\_predicted)

    end\_time = time.time()

    print "runtime = "+str(end\_time - start\_time)+" seconds"

if \_\_name\_\_ == '\_\_main\_\_':

    start\_time = time.time()

    main()

    end\_time = time.time()

    print "runtime = "+str(end\_time - start\_time)+" seconds"