



2017

A Client based email phishing detection algorithm: case of phishing attacks in the banking industry

Edwin Orina Oroko
Faculty of Information Technology (FIT)
Strathmore University

Follow this and additional works at <http://su-plus.strathmore.edu/handle/11071/5616>

Recommended Citation

Oroko, E. O. (2017). *A Client based email phishing detection algorithm: case of phishing attacks in the banking industry* (Thesis). Strathmore University. Retrieved from <http://su-plus.strathmore.edu/handle/11071/5616>

**A Client Based Email Phishing Detection Algorithm:
Case of Phishing Attacks in the Banking Industry**

Oroko Edwin Orina

Master of Science in Information System Security

2017

A Client Based Email Phishing Detection Algorithm:

Case of Phishing Attacks in the Banking Industry

Oroko Edwin Orina

**Submitted in partial fulfilment of the requirement for the award of a Master of Science
Degree in Information System Security (MSc. ISS) at Strathmore University**

Faculty of Information Technology

Strathmore University

Nairobi, Kenya

June, 2017

This dissertation is available for library use on the understanding that it is copyright material and that no quotation from the dissertation may be published without proper acknowledgment.

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

© No part of this dissertation may be reproduced without the permission of the author and Strathmore University

Oroko Edwin Orina

.....

June 2017

Approval

The dissertation of Oroko Edwin Orina was reviewed and approved by the following:

Dr. Bernard Shibwabo,
Lecturer, Faculty of Information Technology,
Strathmore University

Dr. Vitalis Ozianyi,
Lecturer, Faculty of Information Technology,
Strathmore University

Professor Ruth Kiraka,
Dean, School of Graduate Studies,
Strathmore University

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Dr. Bernard Shibwabo, PhD for his valuable guidance and advice throughout the course of the algorithm planning and development. His willingness to offer his effort and time is greatly appreciated.

I would also like to thank I&M Bank for providing facilities and test data, the staff were ready and willing to support me throughout the course of this research.

Lastly, I thank the Almighty God, my parents, and friends for the much encouragement and support I have received during the period of this research.

Abstract

Today, the banking sector has been a target for many phishing attackers. The use of email as an electronic means of communication during working hours and mostly for official purposes has made it a lucrative attack vector. With the rapid growth of technology, phishing techniques have advanced as seen in the millions of cash lost by banks through email phishing yearly. This continues to be the case despite investments in spam filtering tools, monitoring tools as well as creating user awareness, through training of banking staff on how they can easily identify a phishing email.

To protect bank users and prevent the financial losses through phishing attacks, it is important to understand how phishing works as well as the techniques used to achieve it. Moreover, there is a great need to implement an anti-phishing algorithm that collectively checks against phishing linguistic techniques, existence of malicious links and malicious attachments. This can lead to an increase in the performance and accuracy of the designed tool towards detecting and flagging phishing emails thus preventing them from being read by target. Evolutionary prototyping methodology was applied during this research. The advantages are in the fact that it enabled continuous analysis and supervised learning of the algorithm development until the desired outcome was achieved.

This research aimed at understanding the characteristic of phishing emails, towards achieving defence in depth through creation of an algorithm for detecting and flagging phishing emails. In this research, we have implemented a client-based anti-phishing algorithm. The algorithm is able to analyse phishing links, identify malicious email attachments and perform text classification using a Naïve Bayes classifier to identify phishing terms in a new unread email. It then flags the email as malicious and sends it to the spam folder. Therefore the user only gets clean emails in the inbox folder.

Keywords: Phishing, Evolutionary Prototyping, Linguistic Processing Techniques, Natural Processing Language, Classifier, Naïve Bayes Algorithm, Training.

Table of Contents

Declaration.....	ii
Acknowledgement	iii
Abstract.....	iv
Table of Contents	v
List of Figures.....	viii
List of Tables	ix
List of Acronyms	x
Definition of Terms	xi
Chapter 1: Introduction	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Aim	2
1.4 Specific Objectives	2
1.5 Research Questions	3
1.6 Justification	3
1.7 Scope and Limitation	3
Chapter 2: Literature Review.....	4
2.1 Overview	4
2.2 Types of Phishing	4
2.2.1 Visual-Similarity-Based Phishing.....	5
2.2.2 Malware-Based Phishing	6
2.3 Linguistic Processing Techniques.....	6
2.3.1 Natural Language Processing	7
2.3.2 Algorithms used in Text Classification.....	8
2.3.3 Naïve Bayes Algorithm.....	8
2.4 Spam Filtering	10
2.5 Conclusion	11
Chapter 3: Research Methodology.....	12
3.1 Overview	12

3.2 System Development Methodology.....	12
3.3 Requirements Gathering and Analysis.....	13
3.3.1 Data Collection	14
3.4 System Analysis and Design.....	14
3.4.1 System Analysis.....	14
3.4.2 System Design	14
3.4.3 Research Design.....	14
3.5 Implementation	14
3.6 System Debugging/Testing.....	15
3.7 User Evaluation.....	15
3.8 Deliver to User	15
Chapter 4: System Analysis and Design	16
4.1 System Analysis.....	16
4.1.1 System Requirements and Analysis.....	16
4.1.2 Description of the Algorithm	17
4.2 System Design	19
4.2.1 Use Case Diagram.....	19
4.2.2 Use Case Diagram Descriptions	21
4.2.3 Sequence Diagram	26
4.2.4 Data Flow Diagram.....	27
4.2.5 Activity Diagram	30
4.2.6 Class Diagram.....	31
4.2.7 Entity Relation Diagram	32
4.2.8 Database Schema	33
Chapter 5: Implementation and Testing	34
5.1 Implementation	34
5.2 Testing.....	37
Chapter 6: Discussions	45
6.1 Overview	45
6.2 Discussion of Findings.....	45
Chapter 7: Conclusion, Recommendations and Future Works	47

7.1 Conclusion	47
7.2 Recommendations	48
7.3 Future Works	48
References	49
Appendices	53
Appendix A: Code Snippets.....	53
Appendix A.1: Sample Python Stop Words List Code.....	53
Appendix A.2: Sample Python Remove Stop Words Code.....	53
Appendix A.3: Training Dataset of Phishing Key Words in relation to Banking	54
Appendix A.4: Scan Link in using Virus Total API.....	54
Appendix A.5: Acquiring Email Attachment Name for Extension Analysis	54
Appendix B: Turnitin Report	55

List of Figures

Figure 2.1 Text Classification Process.....	7
Figure 2.2 Example of Naïve Bayes application	9
Figure 3.1 Evolutionary Prototyping	13
Figure 4.1 Use Case Diagram	20
Figure 4.2 Use Sequence Diagram.....	26
Figure 5.1 Downloading a List of new Emails	35
Figure 5.2 Extraction of Links from Email Body	35
Figure 5.3 Naïve Bayes classifier code	36
Figure 5.4 Moving malicious email to the Spam folder	36
Figure 5.5 Paperless W2 Phishing Email.....	37
Figure 5.6 Paperless W2 Phishing Email Detected	38
Figure 5.7 Email Body Extract	38
Figure 5.8 Extracted Raw Phishing Email Body	39
Figure 5.9 Threat Email Contents Captured	40
Figure 5.10 Email Body Text passed to Classifier	40
Figure 5.11 Malicious Link Analysis.....	40
Figure 5.12 Payment Alert (PLEASE CONFIRM) 31646 Phishing Email.....	41
Figure 5.13 Email attachment extension acquired.....	42
Figure 5.14 Malicious email attachment mime type identified	42
Figure 5.15 Malicious email details saved in the database	42
Figure 5.16 Malicious email moved to the spam folder	43
Figure 5.17 Malicious email in User's email inbox folder.....	43
Figure 5.18 Malicious email moved to spam folder after analysis	43
Figure 5.19 Already analysed email detected	44
Figure 5.20 Captured Malicious Emails in the Database.....	44
Figure 6.1 Malicious file extensions	46

List of Tables

Table 4.1 Submit Login Credentials Use Case Description	21
Table 4.2 Perform Phishing Email Checks Use Case Description	22
Table 4.3 View and Read Safe Emails Use Case Description	23
Table 4.4 Flag Phishing Email Use Case Description	24
Table 4.5 Save Phishing Email Details Use Case Description	24

List of Acronyms

ASCII– American Standard Code for Information Interchange

DNS – Domain Name Server

IMAP – Internet Message Access Protocol

J48 – Open Source Java Decision Tree Algorithm

HREF – Hypertext Reference

MLP – Multi-Layer Perceptron

NLP - Natural Processing Language

POP3 – Post Office Protocol

SMTP – Simple Mail Transfer Protocol

SVM – Support Vector Machine

URL – Uniform Resource Locator

Definition of Terms

Algorithm – An algorithm is a process or a step by step procedure aimed at solving a particular problem.

Classifier – A set of rules, methods or statistical procedure that identifies to which category an observation belongs based on already trained set of data whose category is known.

Client – The end device, could either be the application or desktop tier.

Bitsquat – A registered domain name with one bit difference on its IP address with reference to another domain.

Linguistic – of or relating to language.

Malware – An umbrella term used to refer to a variety of malicious software.

Phishing – The practice of sending emails purporting to be from legitimate source in order to lure individuals to reveal their personal information such as ids, passwords and credit card numbers.

Spam – Unsolicited or undesired emails.

Training – The process of using content of a known category and creating a classifier on the basis of the known content.

Tokenize – Splitting a string into desired constituent parts.

Chapter 1: Introduction

1.1 Background

The term Phishing originated in the early 1990s during which the mode of hacking then was via phones. The word ‘Phishing’, coined from the word ‘Fishing’ by the then hackers, refers to the malicious act of luring target users to a fake or mimic website through sending them fake e-mails with redirect links and malwares present in form of executable files in email attachments (Sagar, Naresh, & Reddy, 2013; Jakobsson & Myers, 2006). This helps them acquire user’s personal information such as account numbers, passport numbers, nationality, card id numbers, user names and passwords for fraud and theft purposes.

Today, many organizations use email platforms as a formal source of communication. These platforms could either be internally hosted, for example, Microsoft outlook hosted on an internal organization server accessed by everyone in the domain network, or use of external email services not owned by the organization such as Gmail, Yahoo and other search engines. Research by Wombat Security Technologies found out that phishing attacks continue to increase with negative impacts such as malware infections being at 42%, compromising of user accounts at 22% and data loss getting the weight of 4% in the survey of successful phishing attacks (Mackowiak, 2016).

Research indicates that at most 30% of phishing emails are read, making it a preferred attacking vector. With such a higher success rate, use of malicious email attachments and re-direct links have been the top attack mechanisms of choice by the attackers apart from other attack vectors such as web driven attacks, malware download attacks and attacks through network propagation (Verizon, 2016). It is of great importance to note that phishing attacks in organizations cause a lot of damage not only in terms of data loss but it may lead to huge monetary loss. For example, the Carbanak attack, which according to Kaspersky lab stole \$1 billion from close to 100 banks (Cloudmark, 2016).

From an email platform perspective, this research aims at preventing phishing emails from reaching the target user. It provides a client based mechanism which reviews links in phishing emails, checks for any malicious attachments, and applies linguistic processing techniques by looking at the commonly used phishing terms in malicious emails within the banking sector.

1.2 Problem Statement

Bank customers and staff members, especially those in high positions such C.E.Os, are usually the prime targets as regards phishing email attacks (Bank Phishing Scams, 2016). Reports indicate that the top three industries most affected by phishing attacks are Internet Service providers, Finance and Payment Services industries (Dalasta, 2016). Sadly, most people affected by phishing understand largely the need to keep data confidential and away from unauthorized access. Moreover, phishing emails usually resemble the actual legitimate emails sent by the banks.

This leaves the users with only one way of recognizing phishing emails, that is being keen and vigilant on all visual parts of an email such the logos, message and fonts, which becomes hard given that a busy user would at the very least recognize a well mimicked phishing email (Symantec, 2016). Therefore, there exists a need for research on better ways of detect phishing emails and alert the user thereof or prevent the emails from reaching the users.

This research reviewed the key elements of phishing emails within the banking sector. The main aim is to provide a client-based, improved way of phishing email classification, filtering, and preventing the user from opening and responding to the phishing emails.

1.3 Aim

The main aim of this research is to create a client-based email phishing detection algorithm that would detect and prevent bank staff from responding to phishing emails thus helping banks to reduce the number of phishing email attacks.

1.4 Specific Objectives

- (i) To identify the various types of phishing.
- (ii) To investigate linguistic techniques used in phishing emails.
- (iii) To develop and test a client-based algorithm that captures filters against linguistic techniques, malicious links and malicious attachments used by attackers in phishing emails.
- (iv) To validate the accuracy of the algorithm.

1.5 Research Questions

- (i) What are the various types of phishing?
- (ii) What are the linguistic techniques that can be applied to email phishing?
- (iii) How can malicious links be identified in an email?
- (iv) How can malicious attachments be identified in an email?
- (v) How will the accuracy of the algorithm be validated?

1.6 Justification

The formal way of communication within the banking sector between bank and corporates, or bank to customer is mainly through use of emails. With the busy nature of this field and the ever-growing skill of email phishing attackers, it becomes hard to rely on user awareness and training as the major source of equipping users in bid to reduce the number of attacks of this nature (Cloudmark, 2016). Therefore, there is need to enable existent email platforms to detect phishing emails, in as much as attackers come up with different tricks (linguistic techniques) as regards phishing emails with the intention of luring users for malicious purposes (Yasin & Aduhasan, 2016). With time, such a solution will greatly reduce the number of email phishing attacks (Babu, Achanta, Murty, & Swapna, 2012).

1.7 Scope and Limitation

This research was totally focused on the linguistic techniques that are used in banking phishing emails, the various common techniques used to obscure redirecting links to a mimic site and also finding a way of detecting malicious email attachments. The training dataset of the classifier contained only those frequent words derived from a pool of reported banking phishing emails.

Chapter 2: Literature Review

2.1 Overview

This research focuses on phishing email attacks within the banking sector. Generally, phishing attacks take place as follows (Chipuric, 2015; Sagar, Naresh, & Reddy, 2013): The attacker first creates a counterfeit website to masquerade the legitimate website. The attacker proceeds to send numerous spoofed emails to the target client. The emails usually look so authentic to the extent that some of them are hard to differentiate from the legitimate ones at a mere glance. The emails contain messages that intend to convince or lure the target to their plea. The targeted people then receive and open the email, click on the links therein which redirect them to the mimic sites where they unknowingly enter their personal information.

This research focuses on formulating techniques so as to address the prevalent phishing attacks within the banking sector through analysing re-directing links and attachments as well as implementing a robust linguistic technique. In this case, linguistic techniques means having a good text classification technique that captures key words used in the phishing email. The interest is thus not stemming from the number of times a keyword appears (Chandrasekaran, Narayanan, & Upadhyaya, 2006; Hautzer, Helbig, & Schiefer, 1997), such that we can say if the word password appears 10 times then phishing is likely to be the case. The focus is integrating the evidence of probable phishing keywords obtained from text classification together with the analysis of present malicious links and attachments.

2.2 Types of Phishing

Phishing attacks mostly target confidential information such as user names, passwords, social security numbers, passport numbers, credit card numbers, bank account numbers, pin numbers, birthdates, and mother's maiden names among others. Phishing can be categorized into two (Jakobsson & Myers, 2006), namely:

- (i) Visual-Similarity-Based Phishing.
- (ii) Malware-Based Phishing.

2.2.1 Visual-Similarity-Based Phishing

This entails sending large amounts of spoofed emails, asking the recipients to click on embedded links. The hyperlinks, at a mere glance, are usually hard to suspect making it easy for a user to click on them without know the intent. Major researches done regarding phishing via visual similarities are The Intelligent Phishing Website Detection and Prevention System by Using Link Guard Algorithm (Sagar, Naresh, & Reddy, 2013).

The linkguard algorithm majorly focused on the structure of hyperlinks. A hyperlink consists of the universal resource identifier (URL) and the anchor text. The URL or web address is a reference to a specific web resource while the anchor text displays descriptive information about the URL. For example: `Study at Strathmore`. The anchor text, the part that the user sees, is Study at Strathmore, while the value of the href attribute is the URL.

Attackers strive at the fact that the URL is usually not visible to the user unless upon hovering on the button or the anchor text. The anchor text can thus be Study at Strathmore while the attacker has set a different URL value, which ends up redirecting the user to a different location. On this basis, the linkguard research went on to categorize email phishing techniques as:

- i. Attack techniques in which the domain names in the anchor text are legitimate but the URL points to a different web resource.
- ii. Use of dotted or number format on the URL instead of the actual domain name. Organizations mostly issue domain names for access but not their actual private or public addresses.
- iii. Use of encoding schemes, for example, forming links by encoding alphabets corresponding to their ASCII codes or use of special characters such as @ on the anchor text.
- iv. Use of masqueraded URI in the anchor text with added letters but very similar to the URL of the legitimate site, for example: `Click Here`. The above URL seem to be from PayPal, Inc United States but that is not the case.
- v. Attackers take advantage of the vulnerabilities that exists on the targeted websites such as Cross-site scripting on the legitimate website and thereafter use this vulnerability to redirect the users to their phishing sites. For example:

`Click Here` which redirects to <http://201.241.242.7>, the phishing site, due to cross-site scripting vulnerabilities in the culnm.co.ke site.

Phishing based on visual similarities is also the technique used in Bitsquatting attacks. Bitsquatting follows the same attack vector in that it involves the registration of a domain name with one bit different from another domain that is more popular (Dinaburg, 2016). The main problem here is to differentiate between the popular websites domain and the bitsquat domain. Examples of bitsquat domains are aeazon.com, microsmft.com for amazon.com and microsoft.com respectively (Dinaburg, 2016). Search Engine Indexing can also be used to perform phishing based on visual similarities, where the fake web pages with attractive offers created by the attacker get indexed favourably by a search engine, so that a user would stumble upon it (Biju, Chiong, & Seibu, 2005).

2.2.2 Malware-Based Phishing

This type of phishing usually involves the installing of malicious software on the victim's machine. Thereafter, the malware gathers confidential information from the victim (Jakobsson & Myers, 2006; Gudkova, Maria, Nadezhda, & Tatyana, 2016). In this case, the malware does the same job as that of a re-direct-to masqueraded site, upon clicking on the phishing links. This type of phishing incorporates malwares such as key loggers, Trojans via attachments and hosts file poisoning (Akabar, Nukur, & Hartel, 2014).

2.3 Linguistic Processing Techniques

Linguistic processing techniques involves analysing and representing text for purposes of categorization and classification in order to arrive at a particular decision or conclusion based on the findings (Desai, Mukti, & Giyanani, 2014). The main aim is to enable the algorithm to learn based on the predefined logic through continuous training so that it can understand and analyse future events.

An example of a linguistic processing technique commonly used is the Natural Processing Language. NPL relies on machine learning to automatically learn by analysing a given set (these could either be a large corpus, like a book, or as small as a collection of sentences), and thereafter making statically inference on the training dataset. In general, the more data analysed, the more accurate the model becomes (Kiser, 2016).

2.3.1 Natural Language Processing

Natural Language Processing consists to a set of techniques, which are helpful in solving text related problems (Kaggle, 2015). A document consists of a group of words ordered in sequence conveying a desired meaning. We can be interested in the order of words, the sequence followed, the number of words and the count of particular words in the referred set such that in the end we derive a desired set commonly referred to as a bag of words (Chiang, 2015; Bird & Loper, 2009).

However, the process of classification and categorization of text takes place in two stages namely: the Training stage and the Prediction stage (Sharma, 2015). During the training phase, an extractor transforms each input e.g. email content into a feature set. The resulting features are the baseline for basic information about inputs, each to be used for categorization. They may include extracted words from phishing emails such as Account, Send and Click Here, which are the references for categorizing emails. The pairs of feature sets and labels such as the keywords are the fed into the machine-learning algorithm to produce a model.

Afterwards follows the prediction phase (Sharma, 2015). It entails use of the same extractor to transform unobserved inputs to feature sets that are fed into the model to produce predicted labels.

Figure 2.1 shows the text classification process:

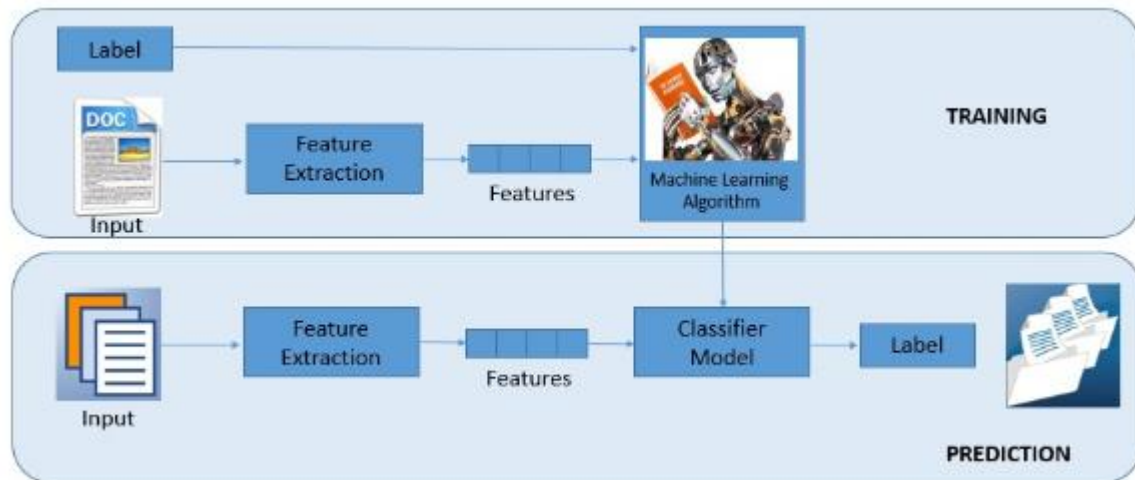


Figure 2.1 Text Classification Process (Adapted from Sharma, 2015)

2.3.2 Algorithms used in Text Classification

As mentioned earlier, there are several algorithms used in text classification to achieve a classifier model. They include Random Forest, J48, SVM, MLP, K-Nearest Neighbour and Bayes Net algorithms. We however look into Naïve Bayes algorithm because:

- i. It is a probabilistic classifier based on the Bayes' Theorem with strong (naïve) independence assumptions between the features (Saed, 2017).
- ii. It requires only a small number of training data to estimate the parameters for classification.
- iii. It performs well in case of categorical input variables compared to numerical variables (Sunil, 2015).
- iv. This research does not entirely depend on the outcomes of the classifier as much as it would achieve classification to a great deal. It also looks into the integration of the classifier results with phishing links and other phishing email components such as available email attachments, in order to finalize its evaluation.

2.3.3 Naïve Bayes Algorithm

Naïve Bayes Algorithm is a conditional probability type of classifier under the Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions related to the event. For example, if phishing emails are arrived at due existence of phishing email keywords, then a particular keyword can be used to more accurately assess the probability that a particular email is indeed a phishing email, compared to the assessment of the probability of phishing emails made without considering that particular keyword.

Bayes' theorem:

$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, Where A and B are events and $P(B) \neq 0$. $P(A)$ and $P(B)$ are the probabilities of observing A and B independently. $P(A|B)$ Represents conditional probability, which is the probability of observing event A given that B is true. $P(B|A)$ represents the probability of observing the event B given that A is true.

Using Naïve Bayes Algorithm, on a given classification problem $x = (x_1, \dots, x_n)$ where n represent the number of independent features, it assigns to this instance probabilities $p(C_k | x_1, \dots, x_n)$ for each of K possible outcomes or classes C_k :

$$p(C_k | x) = \frac{p(C_k)p(x | C_k)}{p(x)}$$

Using Bayes' theorem (Saed, 2017), we result get:

posterior = $\frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$ Where:

- (i) Posterior probability $p(C | x)$, is the probability of the target C (in our case could be the probability of phishing email) given the predictor x (a Keyword fed to the classifier).
- (ii) $p(C)$ - The prior probability of the target.
- (iii) $p(x | C)$ - The likelihood i.e. the probability of the predictor given the target.
- (iv) $p(x)$ - The probability of the predictor.

Figure 2.2 illustrates an example involving suggestions of playing golf given a particular weather. We calculate the posterior probability first by constructing a frequency table for each attribute against the target. Secondly, we transform the frequency tables to likelihood tables and finally use the Naïve Bayes equation to calculate the posterior probability of each class. The class with the highest probability is the outcome of the prediction (Saed, 2017).

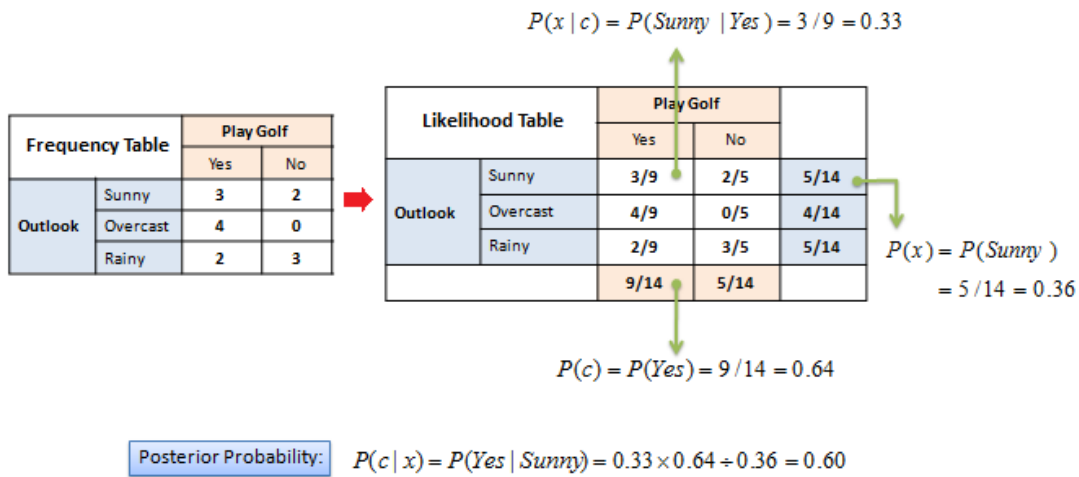


Figure 2.2 Example of Naïve Bayes application (Adapted from Sayad, 2017)

2.4 Spam Filtering

Spam entails the flooding of the internet with many copies of the same email with the aim of forcing the email to reach people who would ideally not choose to receive it. Information about the target is acquired from stealing mail lists or searching target addresses on the web (Thakur, 2017). Mail servers are usually protected from spam emails through configuration of spam filters, creation of rules such as blocking bulk email sending and querying of anti-spam records of the sender server or prior server during the email transmission.

Spam filters are applications installed and configured along with the mail servers to filter against spam emails. They enable one to configure certain filtering rules or decide what activities should be filtered or not. As per their setting they can be viewed as a subset of Spam engines which work at transport level normally installed on server hosting hub. Examples of spam engines are McAfee (McAfee, 2017) and Microsoft Forefront Endpoint Protection for Microsoft Exchange.

Here, all incoming emails pass through Microsoft Forefront first. The malicious identified emails can either be dropped or quarantined while the clean emails get forwarded to Microsoft Exchange (Microsoft, 2011). Microsoft exchange queues emails and authenticates to the active directory before forwarding the emails to the respective users. Other spam engines such as Anti-Spam engine access the user account via the web hosting control panel called cpanel. Some of the features of an anti-spam engine include virus and worm scanning, outbound message filtering, secure message delivery via transport layer security and quarantine management.

Examples of spam filters include SpamAssassin and MailScanner which can be installed together with antivirus applications such as ClaimAV (Mathews, 2017). Spam filters check the source of the message, the software used to send the message and finally the body message content. This entails looking for words frequently used in spam emails such as Click Here, Free, and Now among others. When the above conditions which trigger a particular score are met, for example email has a positive score of two, the spam filter locks the message and sends it to the spam folder.

Most spam emails are usually sent with forged addresses thus email servers lookup the sender domains to establish and validate the sender as either spam or genuine. This is made possible through use of anti-spam records which consist of Sender Policy Framework, reverse Domain Name Server records, Domain Keys Identified Mail and Server reputation.

The Sender Policy Framework available in the server enables domain owner to add a file on the server specific to his domain name thus indicating ownership of the same upon performed lookups. The reverse DNS enables the hostname of the mail server to map to its IP address while Domain Key enables verification of the sender's domain (Mathews, 2017). Spam emails can also be blocked through checking against the mail server blacklist. An example of mail server blacklist is SpamloP. Spam emails can also be prevented at server level by blocking of the spamming IP addresses using a firewall.

However, an attacker can use a real mailer or public mail server such as Amazon, Microsoft, and Gmail in attempt to trick the spam filter. This is due to the fact that blocking of the IP addresses of the public mail server in attempt to prevent spamming would prevent communication by other legitimate users on the same domain. It is not effective as the attacker can use another public mail server to send the same spam emails again. In bid to trick spam filters the attacker can embed links on images or even use homographs and eventually end up bypassing the email filter (Osman, 2014).

Moreover, web mails such as Gmail have implemented antivirus scanners which scan attachments at the time that they are being uploaded. Unfortunately, these scanners normally check for viruses only (ComboFix, 2017). They are also prone to false positives. Previous research on spam mail filtering techniques using different decision tree classifiers have been done with the aim of identifying how accurate text classification can be as regards spam detection. The conclusion was that high level of accuracy can be achieved with a properly justified scope of improvement in terms of false positives (Sarit & Mondal, 2012; Yasin & Aduhasan, 2016; Babu, Achanta, Murty, & Swapna, 2012).

2.5 Conclusion

Based on the literature reviewed, the conclusion is that with spam protection at server level as well as the incorporation of spam filters, users still get phishing emails. Therefore, there still exists a gap and need for in-depth defence which can be achieved through the use of a client-based phishing detection algorithm. The algorithm would base its analysis on an integrated approach, by collectively looking at the linguistic techniques used by the attackers, performing link analysis and analysing email attachments of all new unread emails that reach the client. This will go a long way in making sure the client only gets clean and safe emails in their inbox email folder.

Chapter 3: Research Methodology

3.1 Overview

In this chapter, this research explains the selected research methodology, the reasons why that methodology is preferable, the research design, data collection and analysis techniques used. The chapter comprises of the following key aspects: system development methodology, requirements gathering and analysis, data collection, data analysis and ethical considerations.

3.2 System Development Methodology

One of the best methodologies to use for building adaptive machine learning systems is the rapid/evolutionary/cyclical Prototyping (Neil & Brewerton, 2005). The Evolutionary development methodology enables an early and quick approximation of the final product. After initial development, the prototype is tested and thereafter reworked as necessary to meet the required expectations.

This research adapted the Evolutionary Prototyping Methodology. The methodology has six main development phases. They consist of Requirements Gathering and Analysis, System Design, Implementation (Build Prototype), System Debugging/Testing, User Evaluation, and Deliver to user (Fadi, 2006). Figure 3.1 shows the repetitive nature of each phase during system development:

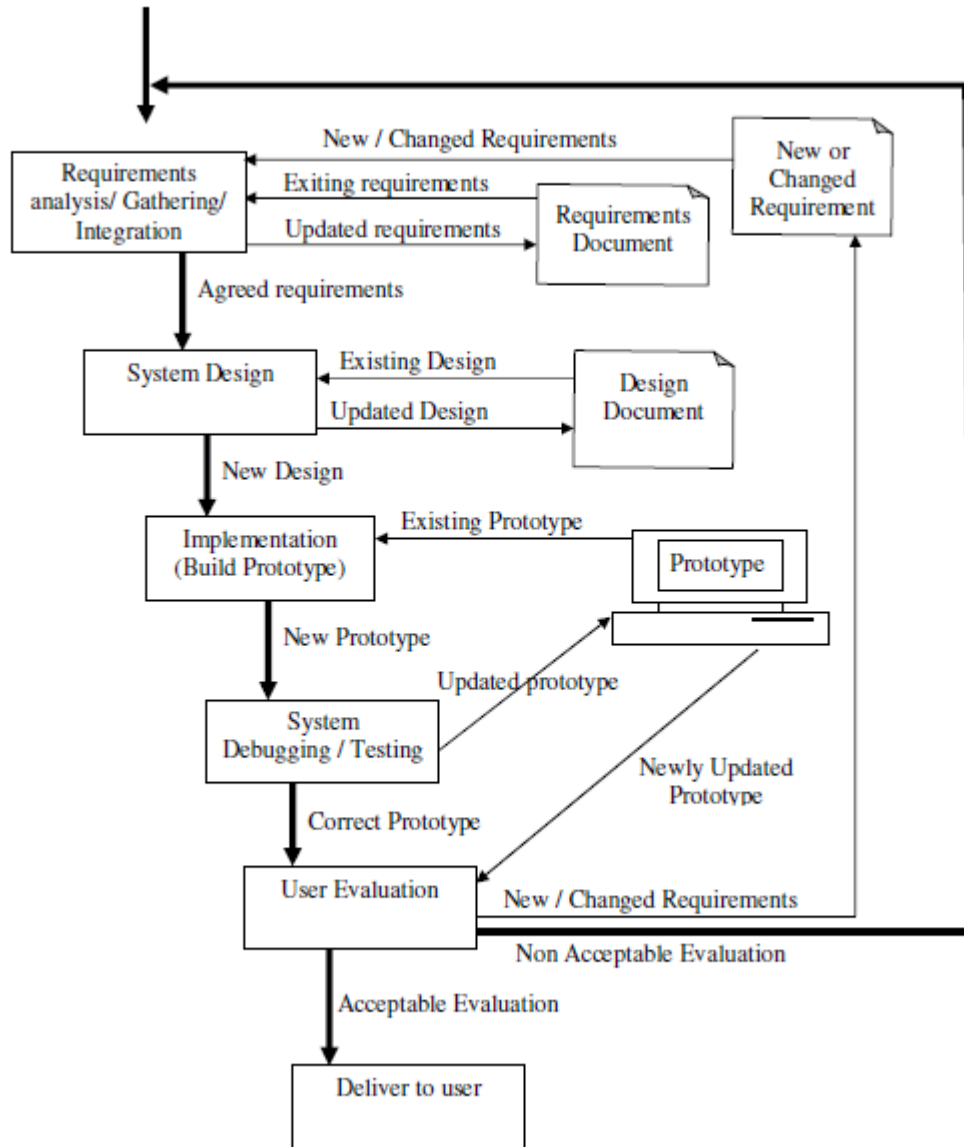


Figure 3.1 Evolutionary Prototyping (Adapted from Fadi, 2006)

3.3 Requirements Gathering and Analysis

This phase was covered to greater extend in the first two chapters when coming up with the objectives and proceeding to do the literature review. This enabled a clear definition of the problem and assessing whether the use of an adaptive system was most appropriate in achieving the set objectives.

3.3.1 Data Collection

Data collection involved acquisition of phishing emails for implementing and testing the classifier. The phishing emails were acquired from the bank's email server and online organizations that have storage of reported banking phishing email attacks such as Berkeley Information Security and Policy (Open Berkeley, 2017) and Phish Tank.

3.4 System Analysis and Design

System analysis and design involved the following:

3.4.1 System Analysis

This entailed coming up with the conceptual design. It also involved capturing all the key process aspects that the algorithm needed to look at while identifying a phishing email. This was in line with the findings from the data collection of various phishing emails. It was helpful in designing the algorithm, making sure it met the requirements needed to prevent phishing emails from reaching the target.

3.4.2 System Design

The design phase was as per the results of the analysis. It will entailed coming up with the use case diagram, sequence diagram, class diagram and the entity relation diagram.

3.4.3 Research Design

The purpose of this research is to develop an adaptive anti-phishing algorithm for banking. To come up with a well-functioning algorithm, previously done algorithms are analysed to form a basis of research requirements (Okstate, 2017). This research employed Quantitative research techniques to this regard. It helped in supervised learning through use of Naïve Bayes theorem to create a classifier, updating it from time to time, in order to improve on the accuracy of the algorithm.

3.5 Implementation

As mentioned above and shown in Figure 3.1, this research employed evolutionary prototyping. The programming language used for algorithm development was Python 2.7. This is due to its simplistic structure and support for large programs (Prasad, 2016). It has also been used previously to develop machine learning programs thus availability of plenty learning resources (Richert &

Coelho, 2013). This research used Portable SQLite for database development. Both Python and SQLite are open source utilities, which are fully compatible (Python, 2017).

3.6 System Debugging/Testing

This phase of the research involved installing, deploying, demonstrating and testing using pilot and stress testing. It also involved checking and improving most of the non- functional requirements, for example, ensuring the system is user friendly and accurate in terms of the results. It also involved continuous programming and debugging until the desired outcome was realised.

3.7 User Evaluation

This involved testing the performance of the algorithm within the ideal banking environment. Here, we were able to get the banks feedback as regards the algorithm and thus enforced needed requirements and modifications.

3.8 Deliver to User

This phase entails operation, maintenance, upgrading and periodic evaluation of the algorithm performance.

Chapter 4: System Analysis and Design

4.1 System Analysis

This section discusses in detail the data requirements and analysis, functional and non-functional requirements as well as giving a description of the algorithm.

4.1.1 System Requirements and Analysis

In order for the algorithm to capture and prevent phishing emails, it is mandatory to develop the structure considering the malicious techniques present in phishing emails. It is thus from the requirements analysis that one identifies the appropriate resources that will satisfy the needs and requirements based on the objectives set.

According to Enfocus (Solutions, 2017), system requirements is an important process as it enables the proposed system to capture the existing gaps. It also enables easy management of the end solution as well as ensuring that the end product fits the organization structure. However, system requirements requires that each need is broken down and defined clearly. A review of the flow of all events as per the interaction of each requirement is then performed, so as that decision making on whether or not the requirement is needed, becomes easier. System requirements analysis enables creation of a development framework thus providing a good basis for all future works.

We can categorized system requirements into two, namely: Functional and Non-Functional requirements (Solutions, 2017). Functional requirements entail the functions that the system must do or deliver. They are the desired functionality that the client expects from the proposed system. A functional requirement also describes the interaction between the system and its environment.

The functional requirements of the proposed algorithm are as follows:

- i. The algorithm should access the user's email account frequently as per set schedule time, and fetch all new emails. It should then proceed to:
- ii. Extract email details for each new email.
- iii. Perform anti-phishing email checks.
- iv. Store phishing email details.
- v. Flag the identified phishing email in user's email inbox by moving it to the spam folder.

The Non-Functional requirements of the proposed algorithm are as follows:

- i. The algorithm should be as accurate as possible to increase reliability.
- ii. The algorithm should be always available, as a client application. This will ensure that it is always checking the user email account checking through any new email once received.
- iii. The algorithm client should have sufficient internet connections to ensure effectiveness in its performance.
- iv. The algorithm should not require a lot of maintenance.
- v. The algorithm should take very little time possible to perform its analysis given the average number of emails received by the client.
- vi. The algorithm should be accurate, readily available while not interfering with the overall email platform experience in order to increase the user level of trust.
- vii. The algorithm should be able to perform and achieve its requirements based on the variety of checks implemented therein.
- viii. The algorithm should ensure security of the user's emails. It should only delete phishing emails.
- ix. In terms of the user's perspective, the algorithm should work seamlessly. The user should not have a clue of any undergoing anti-phishing analysis.

4.1.2 Description of the Algorithm

The algorithm runs on the client computer as opposed to the case of email scanners that work on the mail server. The database used is SQLite portable thus no centralized storage is used. The portable database only store email details of each and every email analysed, it does not store the emails as they reside on the user's email account. At first, the algorithm authenticates to the email client's account through the use of IMAP protocol. This login is independent and separate from the user's actual log in. Here, it is the algorithm which logs into the user's account. After successful login, it checks the inbox for any new emails and fetches the raw email contents.

The algorithm contains a set of phishing words that are used for training of the classifier. These words include: click, here, attachment, link, agent, pdf, and zip among other that normally appear with the body of many bank phishing emails. This set of words form the class category phish (**C**) and are used for calculating the predictor prior probability $p(x)$. This is done for each term in

the training dataset $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ where n represent the number of independent features in the training dataset.

Using the python library, Imaplib, the algorithm fetches the email body plain text part which contains the links and the written text in the email. It applies regular expressions to remove the encoded whitespace characters as a result of using IMAP protocol. The algorithm hashes this part (the written text and links) to base64 and stores this in the database.

Using regular expressions, all links are separated from the written text. The algorithm then picks the written text, tokenizes it to remove all unwanted symbols in order to remain with words only. It then removes all stop words from this group of words and passes the remaining words to the Naïve Bayes Classifier for calculation of the probability of phishing words existing among them.

At the Naïve Bayes Classifier, it proceeds to calculate the likelihood of each word: $p(\mathbf{x} | \mathbf{C})$ and multiplies it with the probability of it being of class $p(\mathbf{C})$ to derive $p(\mathbf{C})p(\mathbf{x} | \mathbf{C})$. The probability of the email being a phishing email is then derived as the total sum of $p(\mathbf{C})p(\mathbf{x} | \mathbf{C})$ divided by the earlier predictor prior probability $p(\mathbf{x})$ of the training dataset. The algorithm calculates the log of the probability of each term and adds them in order to obtain a negative phish score. This is due to the fact that the log of all numbers between zero and one is a negative value.

The algorithm then proceeds to analyse all the links obtained in the email body. Analysis of the links are done by scanning each acquired link on Virus Total via the available Virus Total API. The API enables one to get scan reports without using their HTML web interface. Virus Total is a subsidiary of Google, is a free online malware, worms, viruses, trojans and other malicious content analyser which has a huge blacklist and a wide range of scanners thus mitigating against false positives that may be the case in using one or a few scanners (Quintero, 2017).

The algorithm then check the email attachments. Here it only picks the extension from the file name and checks against a list of malicious file extensions and mime types. If the attachment is found to be malicious, it adds to the negative reputation score of that email. The algorithm does not extract any attachment, it only gets the filename of the attachment which is present in the Content-Disposition part of every email body as per the RFC822 format.

For each and every analysis done the reputation score is the cumulatively calculated taking into consideration all checks performed. When an email gets a negative score from the classification by the NPL (Naïve Bayes classifier), the algorithm recognizes that it is a phishing email and gives it the reputation score of -5. However, the reputation from the malicious links is -15 and that of the attachments -15 in order to increase the accuracy of the algorithm. A phishing email has to have either a malicious attachment or a phishing link in order for it to be considered as an attack. Thus the algorithm flags all emails with -15 as threats and this thus makes it extremely difficult for the algorithm to flag a legitimate email. Moreover, all emails of the same content (exact copy) sent to different clients will have the same score.

The algorithm uses this analysis as a reference to access the malicious email in the user's inbox, flag it as malicious and then transfer it to the spam folder. The algorithm continuously searches for new emails, repeating the same process of analysis and flagging of phishing emails. When an attacker sends a malicious email again, the algorithm identifies it because its hash already exists in the database. It thus flags it with no need of performing the entire analysis all over again. This is only the case in the event that the contents of the email body do not change. Therefore, the client continues receiving and reading clean emails as usual.

4.2 System Design

This section contains a showcase of use case diagram, use case diagram description tables, sequence diagram, data flow diagrams, activity diagram, class diagram and finally the entity relation diagram.

4.2.1 Use Case Diagram

Use case diagrams describe the behaviour and action sets that the system should be perform while it interacts with one or more external users of the system. The term use cases refers to the set of actions performed by the system. Here the system is the subject while the external users are the actors (UML-Diagrams, 2017). Figure 4.1 shows the use case diagrams for the anti-phishing algorithm.

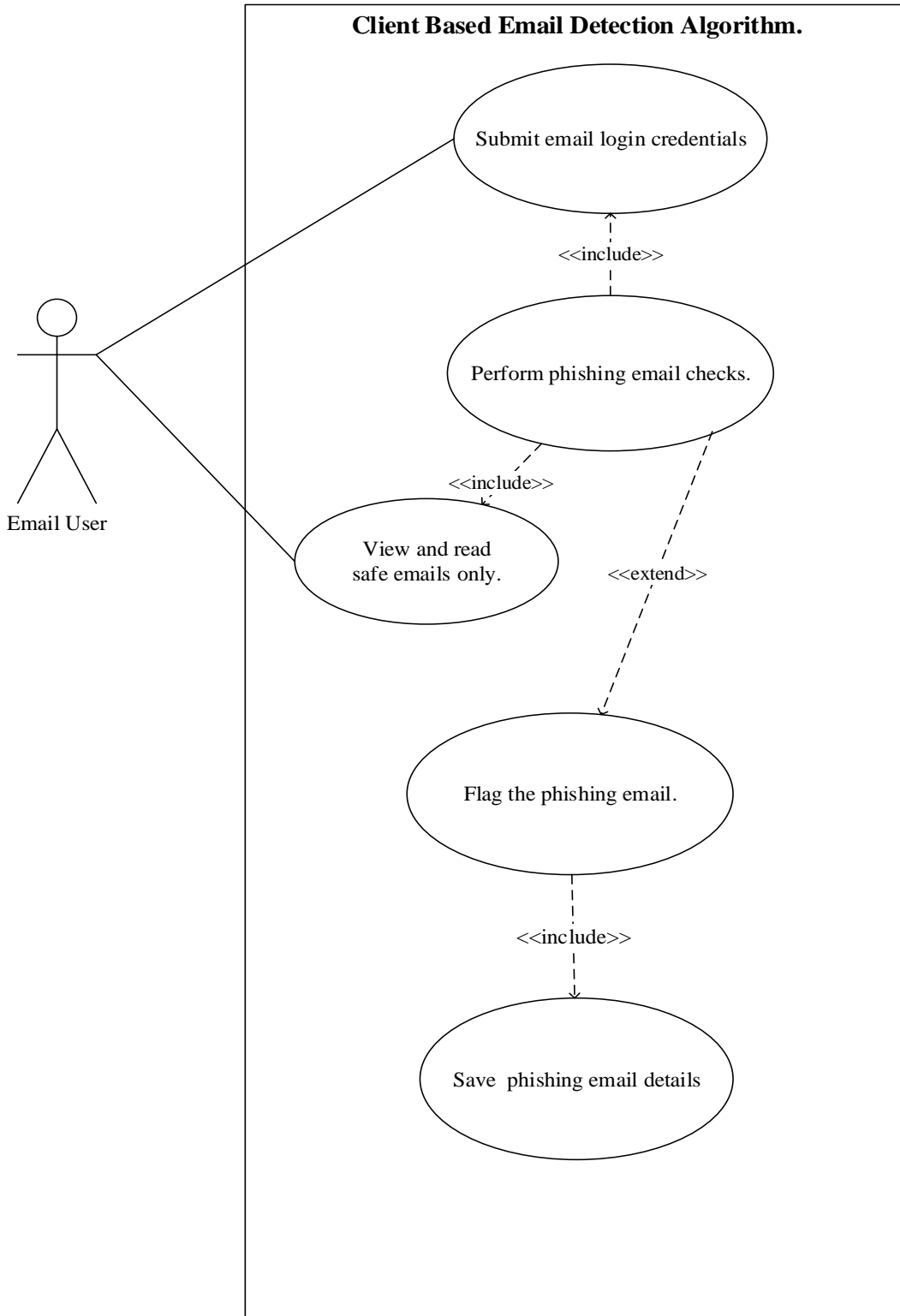


Figure 4.1 Use Case Diagram

4.2.2 Use Case Diagram Descriptions

Use case diagram descriptions are simple explanations of a system's functions from the bird's-eye view of the users (Dennis & Tegarden, 2005). They contain all the information required to produce use case diagrams. They are helpful in that they enable a detailed explanation of each required use case individually. The use case diagrams, for all the use cases developed as shown in Figure 4.1, are as follows:

Table 4.1 Submit Login Credentials Use Case Description

Use Case Name: Submit Login Credentials		ID: <u>1</u>	Importance Level: High
Primary Actor: Email User		Use case type: Overview, Essential	
Brief Description: This use case indicates the algorithm will require user to provide email login credentials so that it can authenticate to the mail server.			
Trigger: The algorithm shall be set to login and continuously check for new emails.			
Relationships Association: Email User.			
Normal Flow of Events: i. The algorithm prompts user to provide their email account username and password. ii. The user provides the access credentials to the user’s email account. iii. The algorithm continuously checks for any new emails and if it finds any it downloads it, ready for the analysis			
Assumptions: i. The user has enabled IMAP authentication to their email account.			

Table 4.2 Perform Phishing Email Checks Use Case Description

Use Case Name: Perform Phishing Email Checks.		ID: <u>2</u>	Importance Level: High
Primary Actor: Algorithm		Use case type: Detailed, Essential	
Brief Description: This use case describes how the algorithm will perform phishing email checks on the new download emails.			
Trigger: Upon checking that there exists a new email for that particular user in the server.			
Relationships Association: Algorithm Include: Submit Login Credentials.			
Normal Flow of Events: i. Obtain email details of any new unread email. ii. Checks if the email hash value (base64 hash) already exists in the database under the threat emails. If it does, it proceeds to move that particular email from the user’s email inbox folder to the spam folder. iii. Extract the email message. Using a Naïve Bayes Classifier, the algorithm performs analysis of the words and provides a score. iv. Extract the links and calculate their reputation based on feedback obtained from blacklists. If malicious, increase phish reputation score. v. Check for any malicious attachments and add up to the phish reputation score. vi. Aggregate the sum of phishing score and provide the email reputation, as clean or as threat email.			

Assumptions:

- i. The algorithm performs anti-phishing analysis only on new emails at the point of authentication.

Table 4.3 View and Read Safe Emails Use Case Description

Use Case Name: View and Read Safe Emails only.		ID: <u>3</u>	Importance Level: High
Primary Actor: Email User		Use case type: Overview, Essential	
Brief Description: This use case indicates the email user, upon authenticating to their email account, will only see clean emails in the inbox folder.			
Trigger: The user logs into their email account as usual.			
Relationships Association: Algorithm, Email User Include: Perform phishing email checks.			
Normal Flow of Events: i. The user logs into their email account as usual. ii. The user’s email inbox is free of any phishing emails. iii. User reads emails as usual.			

Table 4.4 Flag Phishing Email Use Case Description

Use Case Name: Flag Phishing Email.	ID: <u>4</u>	Importance Level: High
Primary Actors: Algorithm.	Use case type: Detailed, Essential	
Brief Description: This use case indicates that the algorithm flags any obtained phishing email after performing the anti-phishing analysis.		
Trigger: Successful analysis indicates that the email is indeed a phishing email.		
Relationships Association: Algorithm, Email User Extend: Perform Phishing Email checks.		
Normal Flow of Events: i. Get phishing email id from database under the threat emails. ii. Refer to that particular email in the user’s inbox and flag it as phish by moving it to the spam folder.		

Table 4.5 Save Phishing Email Details Use Case Description

Use Case Name: Save Phishing Email Details.	ID: <u>5</u>	Importance Level: High
Primary Actors: Algorithm.	Use case type: Detailed, Essential	
Brief Description:		

This use case indicates that the algorithm saves the hash value of any obtained phishing email after performing the anti-phishing analysis.

Trigger:

Successful analysis indicates that the email is indeed a phishing email.

Relationships

Association: Algorithm.

Include: Flag the Phishing Email.

Extend: Perform Phishing Email Checks.

Normal Flow of Events:

- i. Save the full phishing email details to the database including the email hash value. The hash value shall be unique and thus will help identify the same phishing email when sent for second time.

4.2.3 Sequence Diagram

Figure 4.2 shows the sequence diagram, which portrays the interaction among objects during the anti-phishing algorithm operation.

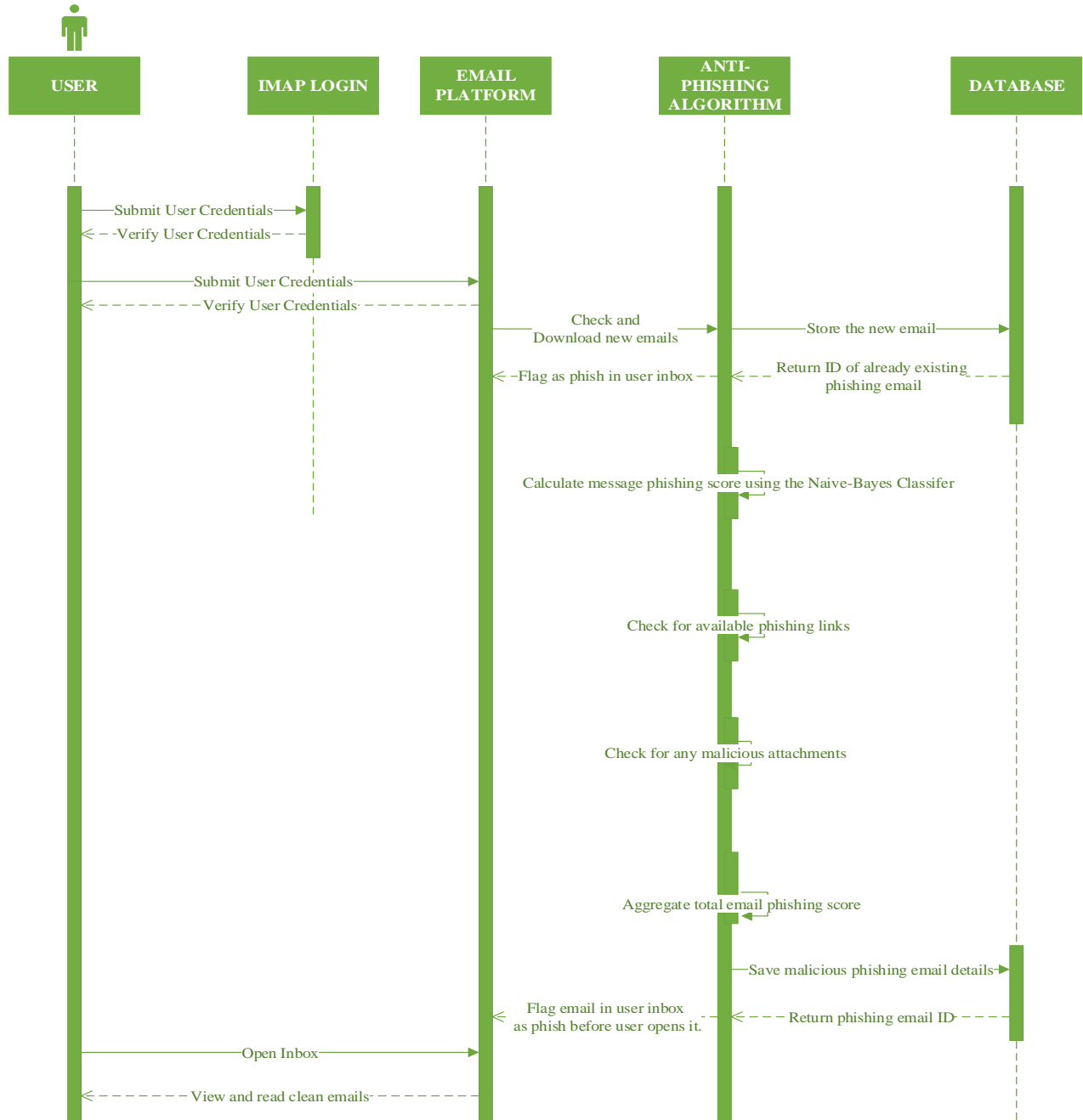


Figure 4.2 Use Sequence Diagram

4.2.4 Data Flow Diagram

Data flow diagrams show the relationship between various components and transformation of data input to output through a sequence of functional requirements. They consist of entities, processes, data stores and data flows (Vie, 2000). The Data Flow Diagrams for the algorithm are as follows:

4.2.4.1 Level 0

Figure 4.3 shows the Level 0 Data Flow Diagram, which portrays an overview of the interaction between the external and internal entities during the algorithm analysis. It also indicates the processes involved and the flow of data between the entities.

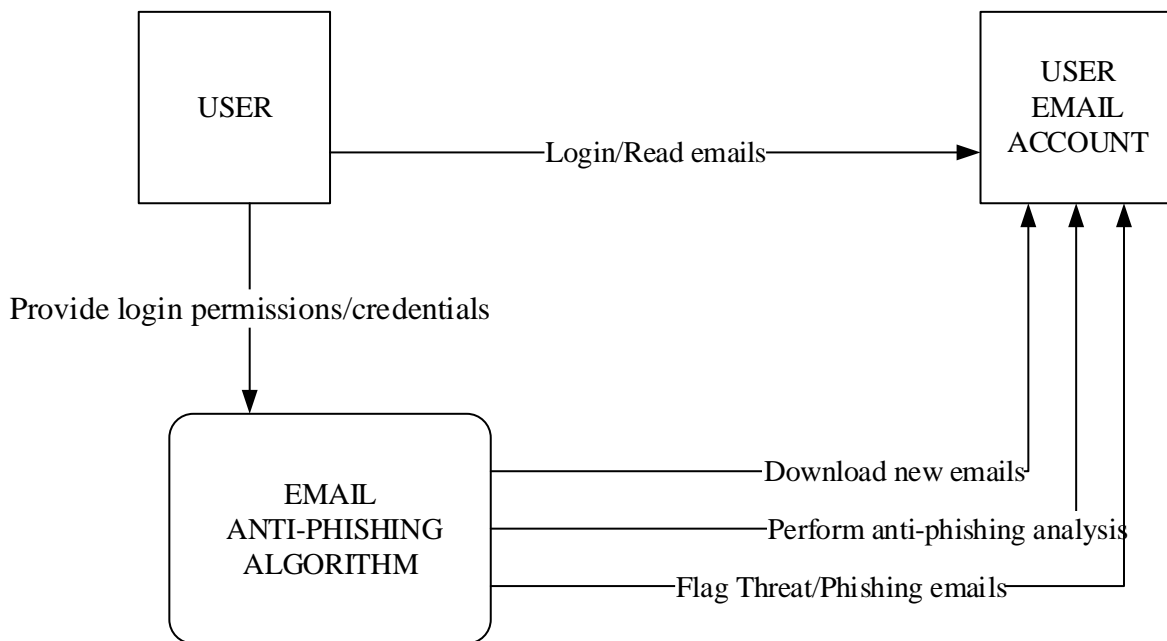


Figure 4.3 Data Flow Diagram Level 0

4.2.4.2 Level 1

Figure 4.4 shows the Level 1 Data Flow Diagram, which portrays the detailed interaction between the external and internal entities during the algorithm analysis.

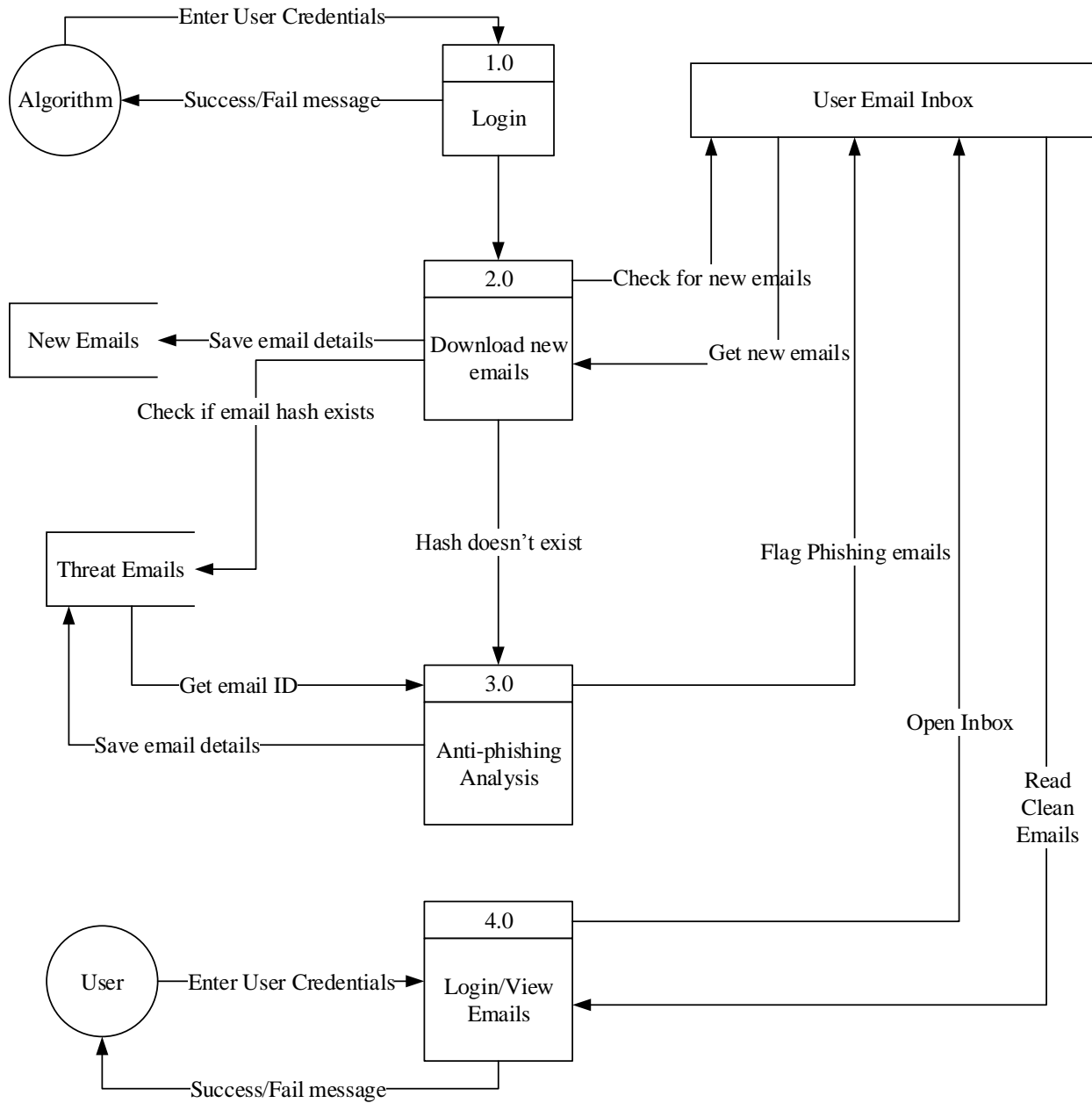


Figure 4.4 Data Flow Diagram Level 1

4.2.4.3 Level 2

Level 2 diagram expounds more on the analysis process, focusing on the core process that must be performed during the algorithm analysis. Figure 4.5 shows the core process that take place between the interaction of both the internal and external entities.

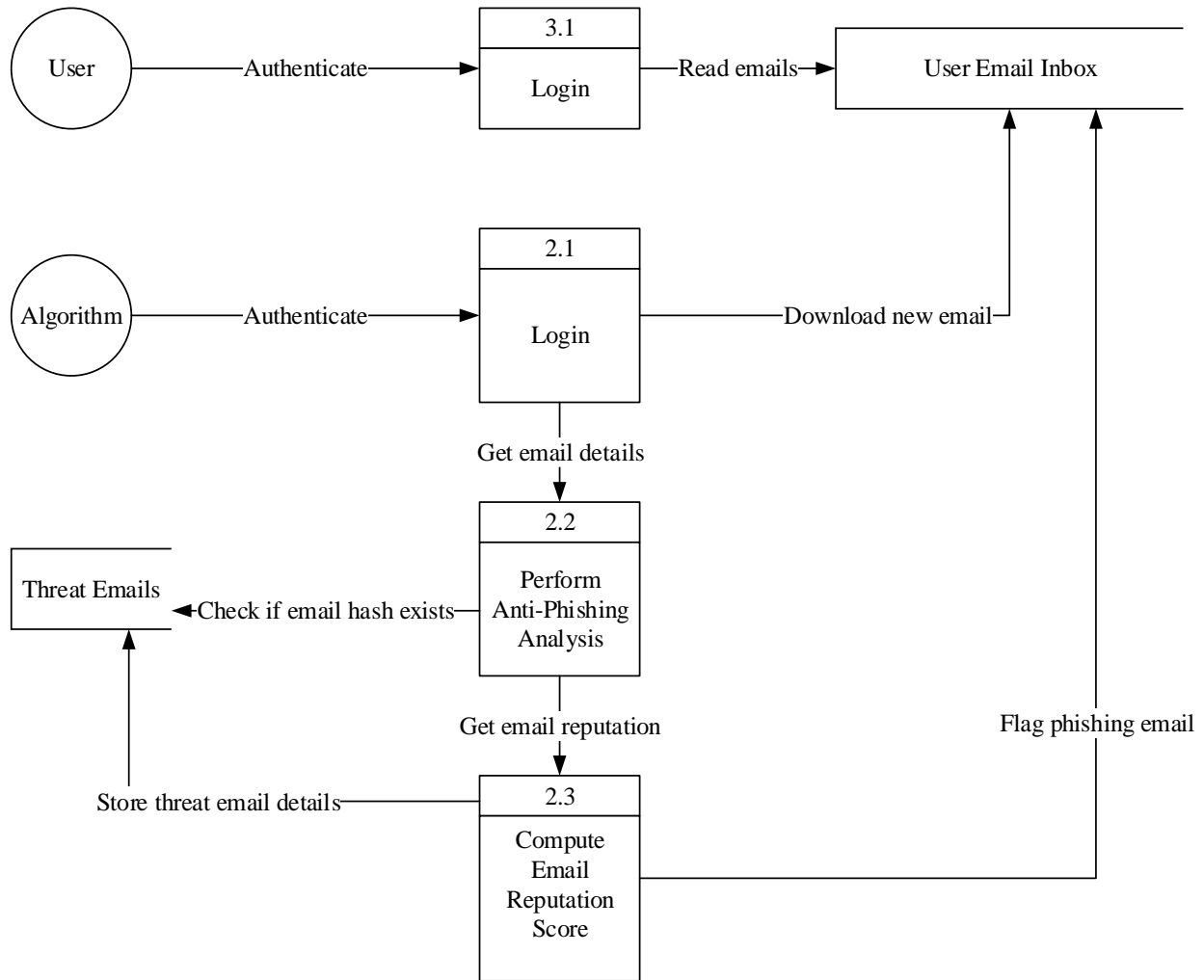


Figure 4.5 Data Flow Diagram Level 2

4.2.5 Activity Diagram

The activity diagrams provide an overview of the entire process flow of control (Dennis & Tegarden, 2005). Figure 4.6 shows the complete process flow applied by the algorithm.

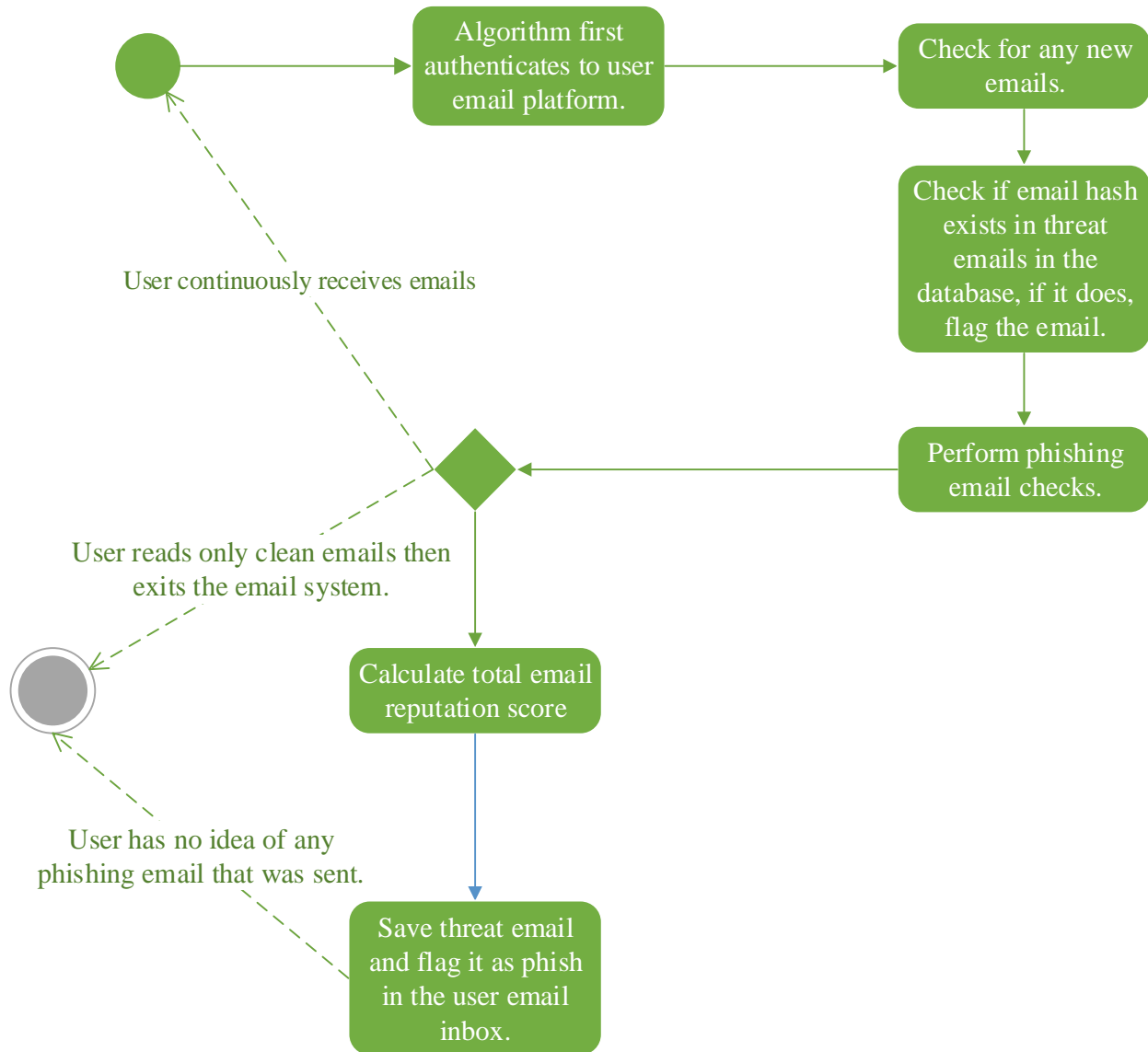


Figure 4.6 Activity Diagram

4.2.6 Class Diagram

A class diagram is a static model that shows the classes and the relationships between them which remains constant in the system over time (Dennis & Tegarden, 2005). Figure 4.7 shows the classes, including both behaviours and states and the relationships between those classes.

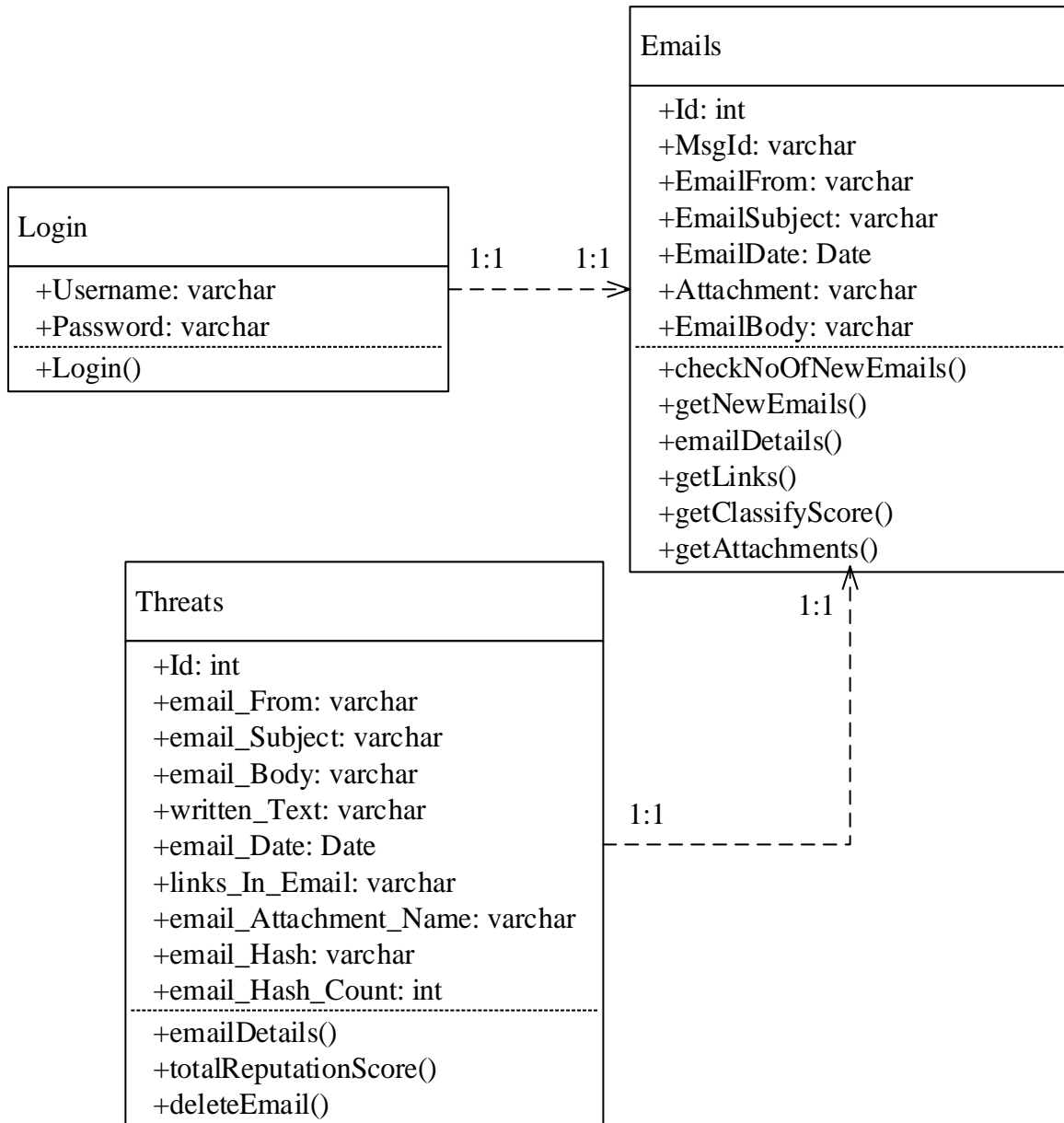


Figure 4.7 Class Diagram

4.2.7 Entity Relation Diagram

Figure 4.7 shows the Entity Relation Diagram. It indicates that one user logs in to see their emails on their email account. Out of the emails there exists a malicious email. There exists a one to one relationship between Login and Emails as well as between Emails and Threats table.

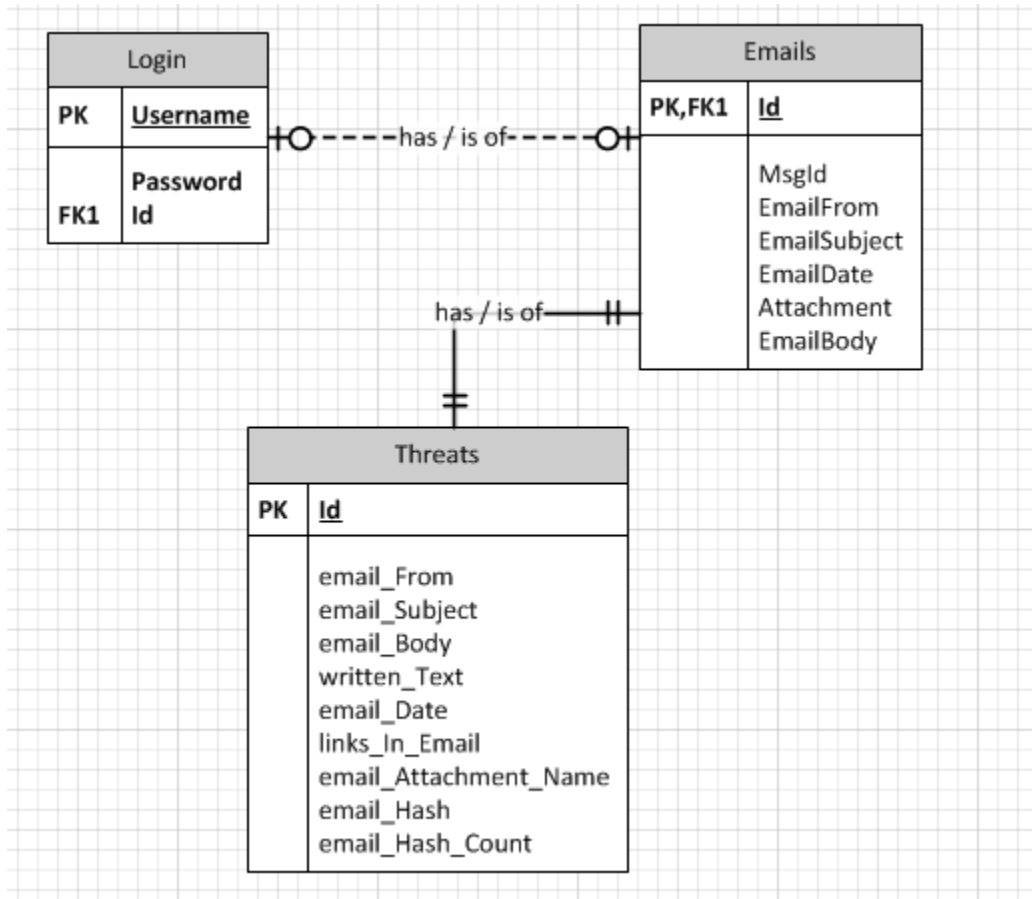


Figure 4.7 Entity Relation Diagram

4.2.8 Database Schema

Figure 4.8 shows the Database Schema.

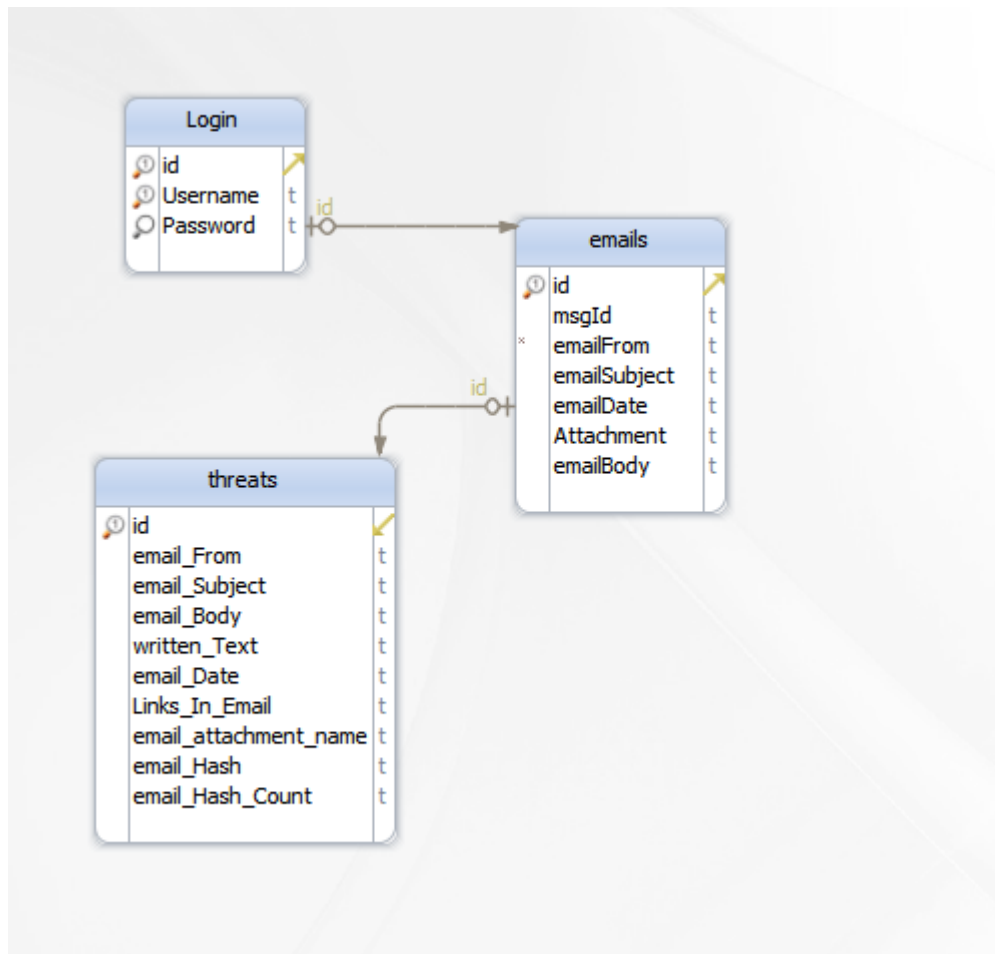


Figure 4.8 Database Schema

Chapter 5: Implementation and Testing

5.1 Implementation

The algorithm development process entails the creation of classes and functions in python code that would enable logging in to the user's email account, performing the anti-phishing analysis and finally flagging any phishing email identified.

For the purpose of classification of phishing emails through text classification by use of Naïve Bayes classifier, email samples from I&M Bank as well as Berkeley Information Security and Policy (Open Berkeley, 2017) were sampled. This enabled a clear understanding of the algorithm requirements for development purposes. The training data fed to our classifier came from analysing the most common words identified within the pool reported phishing email samples. We also used the Google email platform for testing the algorithm. This meets the criteria, as the algorithm only requires IMAP authentication permissions to clients email address.

The classes and functions were implemented using Python programming language while the database was implemented using SQLite database. As discussed in the Literature Review section, the algorithm includes Text Classification using Naïve Bayes Theorem, Extraction and Analysis of email links and identification of any available malicious attachments. These follows after successfully checking whether there exists any new emails, establishing the number of new emails and finally extracting the email details of each new email.

As regards text classification, the classifier contains training data with which it uses to make comparison against email message data. First, it creates a token of each word extracted from the email message body converting it to lower case. Tokenization forms a bag of words to which stop words are removed and the remaining words are later passed to the classifier for Naïve Bayes calculation of phishing probability with reference to training dataset words. The classifier calculates the phish probability score of these words and returns the total phish classification score of that particular email.

Figure 5.1 shows how the algorithm checks for new emails in the user's email account. In the event it finds any new email, it initiates the fetching and extraction process. If there are no new emails, it indicates that there are no new emails found. Here items refers to the id of the email as is in the email inbox. The search function return email response (as result_status whose value is OK) and

the items which is the id of the email. In the event one opens a new email account, the first email received has the id of 1 and this value increments as new emails are received thus identifying them uniquely.

```
result_status, items = server.search(None, "UNSEEN")
items = items[0].split()
global id

if len(items) < 1:
    print "[-] Whoo! No new Emails"
#close the database connection.
    conn.close()
#    exit()

elif len(items) > 0:
    print "[+] We have New %d Emails" % len(items)

    print "[+] Starting Phishing Analysis..."
```

Figure 5.1 Downloading a List of new Emails

As shown in Figure 5.2, the algorithm searches through the entire email body and extracts all the valuable links.

```
def findLinks(body):
    URL_REGEX
    =r'""(?!)\b((?:https?:(?:/{1,3}[a-z0-9%])|([a-z0-9.-\~]+)[. ](?:com|net|org|edu|gov|mil|aero|asia|biz|cat|coop|info|int|jobs|mobi|museum|name|post|pro|tel|travel|xxx|ac|ad|ae|af|ag|ai|al|am|an|ao|aq|ar|as|at|au|aw|ax|az|ba|bb|bd|be|bf|bg|bh|bi|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|cc|cd|cf|cg|ch|ci|ck|cl|cm|cn|co|cr|cs|cu|cv|cx|cy|cz|dd|de|dj|dk|dm|do|dz|ec|ee|eg|eh|er|es|et|eu|fi|fj|fk|fm|fo|fr|ga|gb|gd|ge|gf|gg|gh|gi|gl|gm|gn|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|hr|ht|hu|id|ie|il|im|in|io|iql|ir|is|it|je|jm|jo|jp|ke|kg|kh|ki|km|kn|kp|kr|kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|lv|ly|ma|mc|md|me|mg|mh|mk|ml|mm|mn|mo|mp|mq|mr|ms|mt|mu|mv|mw|mx|my|mz|na|nc|ne|nf|ng|ni|nl|no|np|nr|nu|nz|om|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|ps|pt|pw|py|qa|re|ro|rs|ru|rw|sa|sb|sc|sd|se|sg|sh|si|sj|Ja|sk|sl|sm|sn|so|sr|ss|st|su|sv|sx|sy|sz|tc|td|tf|tg|th|tj|tk|tl|tm|tn|to|tp|tr|tt|tv|tw|tz|ua|ug|uk|us|uy|uz|va|vc|ve|vg|vi|vn|vu|wf|ws|ye|yt|yu|za|zm|zw)/)(?:[^\s()<>{}|\\[\]]+|\\([^\s()<>{}|\\[\]]+)))(?:[^\s()<>{}|\\[\]]+|\\([^\s()<>{}|\\[\]]+)))*[^\s()<>{}|\\[\]]+)'
    finding = re.findall(URL_REGEX,body)
    unique_finds = list(set(finding))
    return unique_finds
```

Figure 5.2 Extraction of Links from Email Body

Figure 5.3, shows the python code implementation of Naïve Bayes classifier.

```
def classify(self, document_input):
    if not self.total_doc_count: raise ClassifierNotTrainedException()

    term_freq_matrix = Counter(document_input)
    arg_max_matrix = []
    for class_id in self.data['class_doc_count']:
        summation = 0
        for term in document_input:
            try:
                conditional_probability = (self.term_count_store[class_id][term] + 1)
                conditional_probability = conditional_probability / (self.data['class_term_count'][class_id] + self.total_doc_count)
                summation += term_freq_matrix[term] * math.log(conditional_probability)
            except KeyError:
                break
        arg_max = summation + self.data['beta_priors'][class_id]
        arg_max_matrix.insert(0, (class_id, arg_max))
    arg_max_matrix.sort(key=lambda x: x[1])
    return (arg_max_matrix[-1][0], arg_max_matrix[-1][1])
```

Figure 5.3 Naïve Bayes classifier code

Figure 5.4 shows how the algorithm flags a threat email in the user's inbox by sending it to the spam folder.

```
def flagEmail(email_id):
    result_status, items = server.search(None, "SEEN")
    items = items[0].split()
    EmailToDelete = email_id
    for EmailToDelete in items:
        server.store("1:{0}".format(EmailToDelete), '+X-GM-LABELS', '\\Spam')
        server.expunge()
    print server.expunge()
    return EmailToDelete
```

Figure 5.4 Moving malicious email to the Spam folder

5.2 Testing

Figure 5.5 shows a phishing email obtained from Berkeley Information Security and Policy. For testing purposes, the email is sent to an email account to which the algorithm has accessed via IMAP. The phishing email, “Paperless W2” (Open Berkeley, 2016), is an example of how CalNet credentials can be compromised. CalNet is an online banking system that enables users to access accounts from anywhere, view and verify transactions, check balances, print statements among other banking related activities. The finding of the email as reported by Open Berkeley include:

- i. The email contains a malicious link, which redirects the user to a counterfeit login page.
- ii. Recipient then enters their CalNet ID and password.
- iii. The attacker now has access to the user credentials.

Original Message:

From: ESSW2@berkeley.edu <huation@clarke.k12.ga.us>

Date: January 6, 2016 at 5:53:32 AM PST

To: undisclosed-recipients;

Subject: IMPORTANT TAX RETURN DOCUMENT AVAILABLE

Dear: Account Owner,

Our records indicate that you are enrolled in the University of California paperless W2 Program. As a result, you do not receive a paper W2 but instead receive e-mail notification that your online W2 (i.e. "paperless W2") is prepared and ready for viewing.

Your W2 is ready for viewing under Employee Self Service. Logon at the following link:

[Click Here](#) to Logon

If you have trouble logging in to Employee Self Service at the link above, please contact your Payroll Department for support.

If you would like to un-enroll in the Paperless W2 Program, please logon to Employee Self Service at the link above and go to the W2 Delivery Choice webpage and follow the instructions.

Figure 5.5 Paperless W2 Phishing Email (Adapted from Open Berkeley, 2016)

Testing was done to confirm whether or not the proposed algorithm can detect the Paperless W2 email on Figure 5.5 as malicious. The account eorina9@gmail.com was used to send the phishing email to another account orinaantiphish@gmail.com to which the algorithm has access to. Upon authenticating, the algorithm detects that a new email has been sent. It then calculates its hash,

having checked that it does not exist in the database and proceeds identify that the email does not have any attachment as shown in the results on Figure 5.6 below.

Figure 5.6 Paperless W2 Phishing Email Detected

```
=====EmailBody=====
--94eb2c08eeb06ef467054de75e49
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

Dear: Account Owner,

Our records indicate that you are enrolled in the University of California
paperless W2 Program. As a result, you do not receive a paper W2 but
instead receive e-mail notification that your online W2 (i.e. "paperless
W2") is prepared and ready for viewing. =E2=80=8B=E2=80=8B

Your W2 is ready for viewing under Employee Self Service. Logon at the
following link:

Click Here to Logon

If you have trouble logging in to Employee Self Service at the link above,
please contact your Payroll Department for support.

If you would like to un-enroll in the Paperless W2 Program, please logon to
Employee Self Service at the link above and go to the W2 Delivery Choice
webpage and follow the instructions.
```

After this phase, the algorithm then displays the raw email body as shown in Figure 5.8.

```
--94eb2c08eeb06ef467054de75e49
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

<div dir=3D"ltr"><p style=3D"box-sizing:border-box;margin:0px 0px 10.5px;color:rgb(51,51,51);font-family:&quot;open sans&quot;;sans-serif;font-size:15px;background-color:rgb(249,249,249)">Dear: Account Owner,</p><p style=3D"box-sizing:border-box;margin:0px 0px 10.5px;color:rgb(51,51,51);font-family:&quot;open sans&quot;;sans-serif;font-size:15px;background-color:rgb(249,249,249)">Our records indicate that you are enrolled in the University of California paperless W2 Program. As a result, you do not receive a paper W2 but instead receive e-mail notification that your online W2 (i.e. &quot;paperless W2&quot;) is prepared and ready for viewing. =E2=80=8B=E2=80=8B</p><p style=3D"box-sizing:border-box;margin:0px 0px 10.5px;color:rgb(51,51,51);font-family:&quot;open sans&quot;;sans-serif;font-size:15px;background-color:rgb(249,249,249)">Your W2 is ready for viewing under Employee Self Service. Logon at the following link:</p><p style=3D"box-sizing:border-box;margin:0px 0px 10.5px;color:rgb(51,51,51);font-family:&quot;open sans&quot;;sans-serif;font-size:15px;background-color:rgb(249,249,249)"><a title=3D"http://ow.ly/WHlx0" style=3D"box-sizing:border-box;background-color:transparent;color:rgb(0,50,98);text-decoration-line:underline">Click Here</a>=C2=A0to Logon</p><p style=3D"box-sizing:border-box;margin:0px 0px 10.5px;color:rgb(51,51,51);font-family:&quot;open sans&quot;;sans-serif;font-size:15px;background-color:rgb(249,249,249)">If you have trouble logging in to Employee Self Service at the link above, please contact your Payroll Department for support.</p><p style=3D"box-sizing:border-box;margin:0px 0px 10.5px;color:rgb(51,51,51);font-family:&quot;open sans&quot;;sans-serif;font-size:15px;background-color:rgb(249,249,249)">If you would like to un-enroll in the Paperless W2 Program, please logon to Employee Self Service at the link above and go to the W2 Delivery Choice webpage and follow the instructions.</p></div>

--94eb2c08eeb06ef467054de75e49--
```

Figure 5.8 Extracted Raw Phishing Email Body

The algorithm then picks out important phishing email content. Figure 5.9 below shows results of the important data derived from the email body.

```

=====
The threat email is:

=====EmailFrom=====
edwin orina <eorina9@gmail.com>

=====EmailSubject=====
=?UTF-8?Q?IMPORTANT_TAX_RETURN_DOCUMENT_AVAILABLE=E2=80=8F=E2=80=8E?=

=====EmailDate=====
Thu, 8 Jun 2017 23:18:16 +0300

```

Figure 5.9 Threat Email Contents Captured

Figure 5.10 shows the results that are returned consisting of the classifier categorization and score. This is after picking out important information and then removing stop words from the email message and passing the remaining written text words to the classifier.

```

['our', 'records', 'indicate', 'that', 'you', 'are', 'enrolled', 'in', 'the', 'university', 'of', 'california', 'paperless', 'w
2', 'program', 'as', 'a', 'result', 'you', 'do', 'not', 'receive', 'a', 'paper', 'w2', 'but', 'instead', 'receive', 'e', 'mai
l', 'notification', 'that', 'your', 'online', 'w2', 'i', 'e', 'paperless', 'w2', 'is', 'prepared', 'and', 'ready', 'for', 'view
ing', 'your', 'w2', 'is', 'ready', 'for', 'viewing', 'under', 'employee', 'self', 'service', 'logon', 'at', 'the', 'following',
'link', 'click', 'here', 'to', 'logon', 'if', 'you', 'have', 'trouble', 'logging', 'in', 'to', 'employee', 'self', 'service',
'at', 'the', 'link', 'above', 'please', 'contact', 'your', 'payroll', 'department', 'for', 'support', 'if', 'you', 'would', 'l
ike', 'to', 'un', 'enroll', 'in', 'the', 'paperless', 'w2', 'program', 'please', 'logon', 'to', 'employee', 'self', 'service',
'at', 'the', 'link', 'above', 'and', 'go', 'to', 'the', 'w2', 'delivery', 'choice', 'webpage', 'and', 'follow', 'the', 'instru
ctions']
[+] Removing Stopwords....

=====DATA TO CLASSIFIER=====

['paperless', 'webpage', 'ready', 'paper', 'result', 'go', 'follow', 'click', 'service', 'notification', 'self', 'please', 'ind
icate', 'program', 'online', 'employee', 'instead', 'payroll', 'here', 'prepared', 'delivery', 'records', 'viewing', 'departmen
t', 'california', 'mail', 'trouble', 'choice', 'instructions', 'enroll', 'logging', 'like', 'receive', 'university', 'contact',
'enrolled', 'following', 'link', 'logon', 'support', 'w2']

=====
('phish', -4.774912960575186)

```

Figure 5.10 Email Body Text passed to Classifier

Figure 5.11 shows link analysis results of the link obtained from the email body. This is the link analysis process which comes after the text classification phase. This is the algorithm begins

```

=====
outcome :malicious

MaliciousLink :http://ow.ly/WHlx0

URL :ow.ly

MaliciousLinkIP :['54.183.130.144', '54.183.132.164', '54.67.120.65', '54.67.62.204', '54.67.57.56', '54.183.131.91']

MaliciousLinkIPDetails :{'raw': None, 'asn_registry': 'arin', 'asn_country_code': 'US', 'asn_date': '2013-11-25', 'asn_cidr':
'54.183.128.0/17', 'raw_referral': None, 'nir': None, 'query': '54.183.130.144', 'referral': None, 'nets': [{'updated': '201
3-11-25', 'handle': 'NET-54-176-0-0-1', 'description': 'Amazon Technologies Inc.', 'postal_code': '98109', 'address': '410 Te
rry Ave N.', 'cidr': '54.176.0.0/12', 'emails': ['abuse@amazonaws.com', 'amzn-noc-contact@amazon.com'], 'city': 'Seattle', 'n
ame': 'AMAZON-2011L', 'created': '2013-11-25', 'country': 'US', 'state': 'WA', 'range': '54.176.0.0 - 54.191.255.255'}], 'as
n': '16509'}

=====

```

Figure 5.11 Malicious Link Analysis

The next observation is on the entire analysis process done by the algorithm from the point it detects a new email till it moves a malicious email to the Spam folder for an email that contains a malicious attachment. Figure 5.12 shows a phishing email reported to I&M Bank Information security office. The email was received by the Call Centre team. It was an attempt purportedly targeting the Funds Transfer department with regards to swift payments.

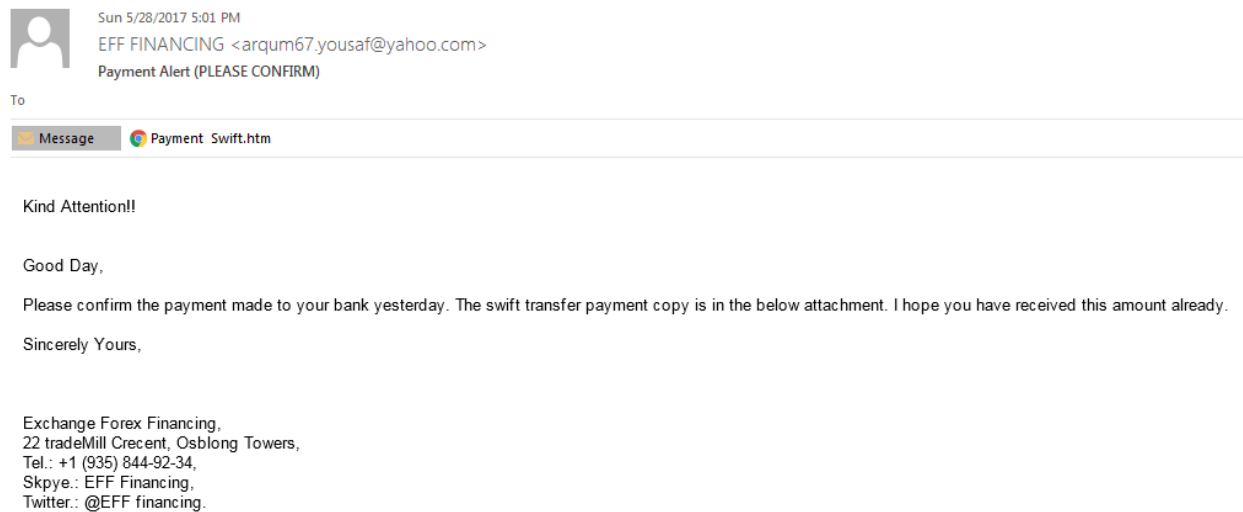


Figure 5.12 Payment Alert (PLEASE CONFIRM) 31646 Phishing Email

On subsequent analysis, it was discovered that the attacker had attached a Trojan called Trojan.Html.PhishAbode.elhpdg to an image in the page of the .html attachment. This would be triggered upon clicking on the image after loading the attached html. There have also been numerous attacks of this nature that have reached the target despite the enforced email filters.

To this effect, the code maintains a list of unwanted file mimes and extensions with which it filters against extensions obtained from the attachment file name. This bears in mind the danger which exists in trying to extract attachments as regards viruses and Trojans as well as the amount of time it would take to scan each and every attachment.

The algorithm analyses the email as follows:

- i. The algorithm logs in, identifies the new email and calculates the hash of the written text in the email body. If the hash value does not exist in the database, it proceeds check for any available attachments. If the attachments exists, it picks the attachment's extension for filtering against the list of malicious extensions as shown in Figure 5.13.

- iv. The algorithm then moves the malicious email to the Spam folder as shown in Figure 5.16.

```
[+] Saving malicious email details to the Database :  
  
[+] Flag By : 1  
[+] Moving malicious email to Spam folder :  
  
('OK', [None])  
[+] Email successfully moved to Spam folder :
```

Figure 5.16 Malicious email moved to the spam folder

- v. Figure 5.17 shows the malicious email in the inbox at 6:45 pm

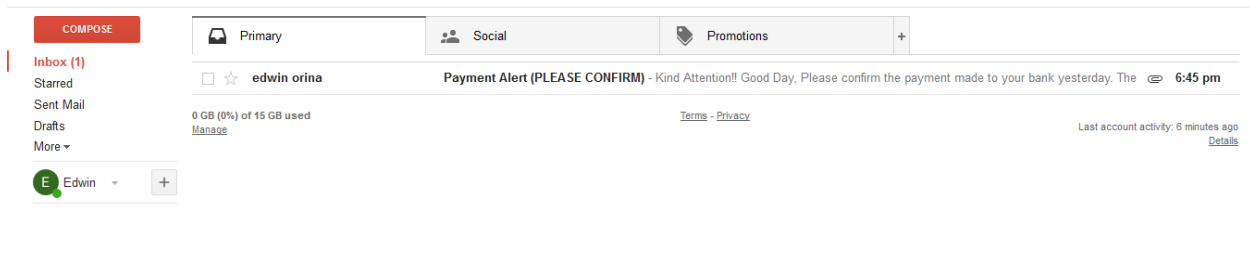


Figure 5.17 Malicious email in User's email inbox folder

- vi. Figure 5.18 shows the malicious email moved from the inbox to the spam folder at 6:45 pm after analysis.

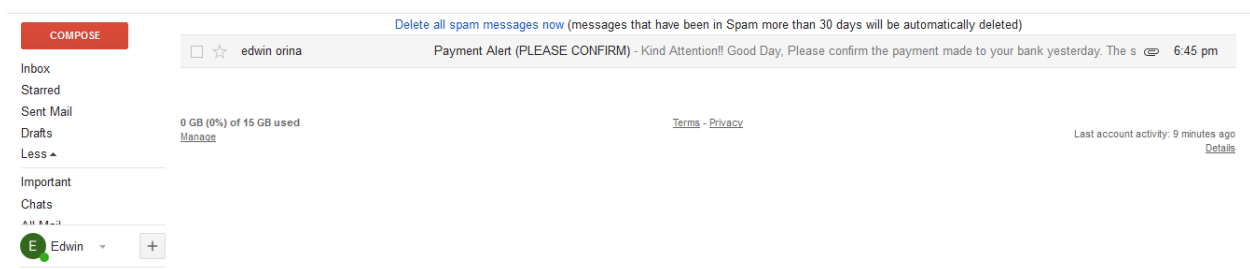


Figure 5.18 Malicious email moved to spam folder after analysis

- vii. On sending the same email again the algorithm check whether the same hash exists. It then moves the email to the trash folder, increments the email_hash_count value of the database and then stops the analysis.

[illegible]

Figure 5.19 Already analysed email detected

threats @ main (Emails) - Table

File Edit View Window Help

Import Wizard Export Wizard Filter Wizard Grid View Form View Memo Hex Image Sort Ascending Sort Descending Remove Sort Custom Sort

id	email_From	email_Subject	email_Body	written_T	email_Date	Links_In_Email	email_attachm	email_Hash	email_Hash_Count
57	edwin orina <orina9@gmail.c	Payment Alert (PLE	--001a11452ea03013bc05514d2b27		Tue, 6 Jun 2017		malicious	DQoNcKtpbmQ:	0
59	edwin orina <orina9@gmail.c	Library Account	--001a11451814975cb205514dda8c		Tue, 6 Jun 2017	http://auth.berkeley.edu.libna.ml/cas/lo	(Null)	DQoNcKRYXJgU	0
61	edwin orina <orina9@gmail.c	RE: Notice	--001a1141e9146434205514e157f		Tue, 6 Jun 2017	http://maintenance.zohosites.com/	(Null)	DQoNcKg0KSGv1	1
63	edwin orina <orina9@gmail.c	=?UTF-8?Q?IMPOF	--001a1141e914447d605514ecb56		Tue, 6 Jun 2017	http://ow.ly/WHb0	(Null)	DQoNcKg0KRGv1	1
65	edwin orina <orina9@gmail.c	URGENT: Your Nev	--001a113ecd00b8e28b05514edb1a		Tue, 6 Jun 2017	http://gabrielramon.be/ http://gabri	(Null)	DQoNcKhlbgxv1	1
67	edwin orina <orina9@gmail.c	Update Western ur	--f403045d9aaa4858f70551590e11		Wed, 7 Jun 2017	http://www.pannonled.hu/js/dist/portal/	(Null)	DQoNcKfmdGV	0
69	Edwin Orina <orinaantiphish@	American Express I	--089e08224db0cfa62a0551591e14		Wed, 7 Jun 2017	http://sweetheale.com/06635/Amx/	(Null)	DQoNcKg0KKRv1	0
71	Edwin Orina <orinaantiphish@	Delivery attempt fa	--001a113ed9e8d91b290551597f83		Wed, 7 Jun 2017	http://www.canadapost.ca/cpotoools/ap	(Null)	DQoNcKg0KRGv1	0
73	Edwin Orina <orinaantiphish@	MailBox is Full	--001a1146df522ef7e055159aa14		Wed, 7 Jun 2017	http://linoflax.com/2017/wp-includes/js/	(Null)	DQoNcKfUUEVC	0
75	Edwin Orina <orinaantiphish@	CallingRemise He	--f403045f2aeec56be2055159de8c		Wed, 7 Jun 2017	https://www.callingremise.com/verifica	(Null)	DQoNcKg0KRGv1	0
77	Edwin Orina <orinaantiphish@	WU System Portal I	--f403045ecc2456241405515a295a		Wed, 7 Jun 2017	ion.com http://www.pannonled.hu/js/	(Null)	DQoNcKfmdGV	0
79	Edwin Orina <orinaantiphish@	Wells Fargo Bank S	--f403045c7d00a6fe2b05515a3f02		Wed, 7 Jun 2017	http://www.ideal-case.com/skin/www.W	(Null)	DQoNcK91ciBW	0
81	Edwin Orina <orinaantiphish@	US\$4.6M TRANSFE	--001a1143f7204796a705515a5948		Wed, 7 Jun 2017	WWW.BKOBKETH.COM http://www.W	(Null)	DQoNcKfUUE4u	0
83	Edwin Orina <orinaantiphish@	SECURITY NOTICE	--001a1148ad943d4c2805515a659f		Wed, 7 Jun 2017	http://infoinfectank.com/ibanking.ban	(Null)	DQoNcKRYXJgU	0
85	Edwin Orina <orinaantiphish@	Message from hun	--001a1143f7204aeb605515a90d4		Wed, 7 Jun 2017	http://www.jsanchez.com/auth.berkele	(Null)	DQoNcKg0KQW4	0
87	Edwin Orina <orinaantiphish@	Your Dropbox File	--001a1146df52b5184b05515aa0c9		Wed, 7 Jun 2017	http://auth.berkeleyedu.atwebpages.c	(Null)	DQoNcKg0KSGv1	0
90	Edwin Orina <orinaantiphish@	Update Portal	--f403045c7d001c9b7805515aa77c		Wed, 7 Jun 2017	prathidwanix.org/www/wz2/	(Null)	DQoNcKlbgxv1	0
92	Edwin Orina <orinaantiphish@	Email Account Upd	--001a1142797cb9e44d05515ab1e8		Wed, 7 Jun 2017	http://goo.gl/rk8TKW	(Null)	DQoNcKRYXJgU	0
94	Edwin Orina <orinaantiphish@	Irregular Activity	--001a113ef68b05ce705515ac1e8		Wed, 7 Jun 2017	http://s21.postimg.org/t6anbf6h3/pic13	(Null)	DQoNcKg0KfDIk	0
96	Edwin Orina <orinaantiphish@	Vital Info	--f403045ecc30d2885405515ad0df		Wed, 7 Jun 2017	https://www.google.com/drive/doc=	(Null)	DQoNcKhlbgxv1	0
98	Edwin Orina <orinaantiphish@	IT Support	--f403045c7d0022bab805515ad488		Wed, 7 Jun 2017	http://bcourses.berkeley.cnea.gg/login	(Null)	DQoNcKRYXJgU	0
100	Edwin Orina <orinaantiphish@	Last Reminder You	--001a1148957a77344605515b0014		Wed, 7 Jun 2017	https://tw.giantleaplab.com/tmp/uri/	(Null)	DQoNcKhlbgxv1	0
102	Edwin Orina <orinaantiphish@	=?UTF-8?B?GVSvc	--001a1146df52482f305515a06b4		Wed, 7 Jun 2017	esksystems.jimdo.com/ http://teamhel	(Null)	DQoNcKg0KfDIk	0
104	Edwin Orina <orinaantiphish@	Your access has be	--f403045dbf64a088905515b2849		Wed, 7 Jun 2017	http://mars1.addhosti= ng.n2g10.com/	(Null)	DQoNcKpEZWfY	0
106	Edwin Orina <orinaantiphish@	Dear Email User	--001a114fc22812a1005515b2f9d		Wed, 7 Jun 2017	ssomd.jimdo.com/	(Null)	DQoNcKlvdXgc	0
108	Edwin Orina <orinaantiphish@	Help Desk / Passw	--94eb2c9d856af6ea05515b36da		Wed, 7 Jun 2017	http://outlookwebaccessform.weebly.c	(Null)	DQoNcKlBhc3N3	0
110	Edwin Orina <orinaantiphish@	Record Update.	--f403045ecc30540d005515b3d2e		Wed, 7 Jun 2017	https://c2.staticflickr.com/4/3163/57904	(Null)	DQoNcKlRoaxMg	0
111	Edwin Orina <orinaantiphish@	Research	--94eb2c1998ec44a7f05515b5617		Wed, 7 Jun 2017	http://www.sciencedirect.com/science/	(Null)	DQoNcKRYXJgR	0
113	Edwin Orina <orinaantiphish@	PO PFM0180/17	--f403045f2aeec507e905515b8798		Wed, 7 Jun 2017	https://tinyurl.com/y83zqf3m http://ov	(Null)	DQoNcKdmb2Qg	0

Figure 5.20 Captured Malicious Emails in the Database

Chapter 6: Discussions

6.1 Overview

This chapter discusses findings of this research in line with the earlier set objectives, research questions and scope, providing an explanation on the key areas covered.

6.2 Discussion of Findings

As per chapter one of this documentation, the key objectives of this research were to be identify types of phishing attacks, investigate on the linguistic techniques, develop and test the client based anti-phishing algorithm. During the literature review, this research looked at the various types of phishing attacks namely, the malware-based and the visual-similarity based phishing attacks. It went on to identify the attack techniques that are used. The techniques include; use of malware attachments, use of phishing links to redirect user to attacker's malicious sites and use of luring language that prompts the user to click the malicious links and open the malicious attachments.

This research settled at handling the phishing email attacks from the client side rather than at server level. Therefore, it was necessary to create an algorithm that would authenticate to the user's email account and continuously check and analyse new emails, flagging any malicious emails identified. Algorithm authentication led to the use of IMAP protocol. The email user first needs to enable IMAP authentication on their email account and provide the login credentials to the algorithm which would then perform the analysis repeatedly as per its scheduled time.

The second objectives of this research focused on the linguistic techniques that are used in the phishing emails. This research was able to identify and detect the phishing words that are mainly used in the banking phishing emails. Appendix A.3 shows how this data was incorporated to create a training data set. Through algorithm reviews during the literature review, this research was able to formulate a solution based on the Naïve Bayes classifier. The algorithm fetches all the new emails, extracts the message body, tokenizes all the words and then tests them against the training dataset to determine the phishing score of that particular email.

The research adds to the quality of email phishing analysis by looking at other aspects of phishing emails other than the luring words. During this research, phishing links in emails were also looked at. In the context of phishing, the anchor link which the user sees, provides a false description of

the link. Therefore, this algorithm extracts all the links in downloaded emails and analyses their reputation as either safe or malicious.

In the event malicious links are found, the algorithm increments the email's phish reputation value. Extraction of the links is important as it is not restricted by the nature in which the link is hidden. For instance, phishing links can be embedded on photographs which the user sees and clicks unaware of the intent. The algorithm also checks for a set format of attachments such as .exe, .msi, .vb, .lib, .bat, .cmd among other file extension types used in malware attacks.

```
extension_blacklist = [  
    'ade', 'adp', 'bat', 'chm', 'cmd', 'com', 'cpl', 'exe', 'hta', 'ins', 'isp',  
    'jse', 'lib', 'mde', 'msc', 'msp', 'mst', 'pif', 'scr', 'sct', 'shb', 'sys',  
    'vb', 'vbe', 'vbs', 'vxd', 'wsc', 'wsf', 'wsh', 'htm', 'html']
```

Figure 6.1 Malicious file extensions

The final phishing email reputation is a result of analysis of the email phishing words, the existence of phishing links as well as the existence of any malicious attachments. All details of the threat/malicious emails are stored in the database together with a hash value of the malicious email. This is a hash of the written words plus the links within the email body to base64.

The algorithm uses the hash value to identify a previously sent phishing email and flag it as malicious without need of performing the analysis again. The final objective of this research was to test the accuracy of the algorithm. As shown in chapter five, various tests were done on the algorithm. The algorithm proved easy to use and reduces the effect of phishing attacks through flagging of the phishing emails.

Chapter 7: Conclusion, Recommendations and Future Works

7.1 Conclusion

Extensive work has been done in this research relating to the Python Programming Language, Email Architecture, Phishing attack techniques and the Naïve Bayes Theorem application in Text Classification. The process of research has been quite intensive and thus bringing out more on phishing and related algorithms which can be even of more use in future research. Much has also been done to integrate the phishing detection algorithm in a real world setting. The end product is useful for organizations and individuals who seek to be secured from malicious email attacks.

The major challenge that was faced during this research was identifying the best context in which the algorithm would work and how it would be able to integrate many users. It also follows that different email platforms use different protocols to receive emails. The most common ones are POP3, IMAP and SMTP.

Moreover, the use of online Virus Total to perform testing was limited to a number of tries that one is allowed to make per minute. The sending of phishing emails to a particular account, for example a Gmail account, to see how the algorithm behaves in a real world setting was difficult due to the fact that those domains are owned thus there no exclusive permission to test against certain nature of phishing email attacks.

However, the algorithm proved useful as it was successfully able to send malicious emails to the spam folder after analysis of the same. These emails had bypassed all the controls at server level and managed to reach the user inbox folder. This research also clearly demonstrated the need to focus on the three main aspects of phishing namely; the linguistic techniques, link analysis and finally malicious attachment analysis. Both three aspects are fundamental towards detection and flagging of malicious emails.

The approach used in this research demonstrates the importance of defence in depth as regards email phishing attacks. It also proved easy to use as the user would only require to enter their login credentials to the plug and play client based email phishing detection algorithm. Therefore, if adopted and integrated with the various email platforms, the algorithm would go a long way in preventing user from getting phishing emails in their email inbox folder thus mitigating the threats resulting from phishing emails.

7.2 Recommendations

The finding of this research were a success in bid to analyse new unread emails once they arrive in the user's email inbox folder, flag all phishing emails found and send them to the spam folder. The algorithm gave positive result as per the subsequent tests carried out. However, the research felt there is need to have more features and made the following recommendations:

- (i) It is important for email platforms to implement Single Sign-On and third factor authentication as is the case for Gmail email platform. Single Sign-On enables access to multiple independent software components that are related on one single login instance. This would enable guaranteed security by allowing login to email account via applications such as the algorithm access while ensuring verification via third party authentication. Here the user allows the algorithm to login to their account, upon which they will be notified of the any other login attempt they have allowed.
- (ii) Due to the large number of emails received by some users, there is need to increase the speed of phishing analysis while maintaining accuracy. This can be achieved through adaptive machine learning.

7.3 Future Works

The future improvements on the Anti-Phishing Email Algorithm include:

- (i) Creating a user friendly plugin that would enable the user to provide access credentials thus ease in running the algorithm on client side browser. This will involve creation of email application specific plugins to enable integration of applications such as Outlook and Thunderbird with the algorithm. This will also enable continuous updates on the algorithm to be done easily.
- (ii) As regards the Naïve Bayes classifier, optimizing term frequency estimation would result to a greater percentage of accuracy of the algorithm in some classification cases. For instance, the number of non-phish words can be equal to the number of phish words in a phishing email message.
- (iii) Creation of enhanced ways of detecting malicious attachments.
- (iv) Applying machine learning techniques to make the algorithm adaptive in nature.

References

- Akabar, Nukur, & Hartel. (2014). Analysing Persuasion principles in phishing emails.
- Babu, Achanta, Murty, & Swapna. (2012). Development of Maize Expert System Using Ada-Boost Algorithm and Naive Bayesian Classifier. *International Journal of Computer Application Technology and Research*, 89-93.
- Bank Phishing Scams*. (2016). Retrieved from Phising.org: <http://www.phishing.org/scams/bank-phishing/>
- Biju, Chiong, & Seibu. (2005). Analysis of Phishing Attacks and Countermeasures. *Analysis of Phishing Attacks and Countermeasures*.
- Bilgi. (2012). *O8-chapter 3.pdf - Shodhganga, Development of Prototype*. Retrieved from [shodhganga.inflibnet.ac.in: http://shodhganga.inflibnet.ac.in/bitstream/10603/5651/8/08_chapter%203.pdf](http://shodhganga.inflibnet.ac.in/bitstream/10603/5651/8/08_chapter%203.pdf)
- Bird, S., & Loper, E. K. (2009). *Natural Language Processing with Python*.
- Chandrasekaran, Narayanan, & Upadhyaya. (2006). *Phishing E-mail Detection Based on Structural Properties*. Retrieved from albany: <http://www.albany.edu/iasymposium/proceedings/2006/chandrasekaran.pdf>
- Chiang. (2015, January 20). *Test Classification*. Retrieved from [www3.nd: http://www3.nd.edu/~dchiang/teaching/nlp/2015/notes/chapter1v2.pdf](http://www3.nd.edu/~dchiang/teaching/nlp/2015/notes/chapter1v2.pdf)
- Chipuric. (2015, December 23). *The ABDs of Detecting and Preventing Phishing*. Retrieved from Heimdal Security: <https://heimdalsecurity.com/blog/abcs-detecting-preventing-phishing/>
- Cloudmark. (2016, January 13). *Survey Reveals Spear Phishing as a Top Security Concern to Enterprises*. Retrieved from Cloudmark Security Blog: <https://blog.cloudmark.com/2016/01/13/survey-spear-phishing-a-top-security-concern-to-enterprises/>
- Dalasta. (2016). *Phishing Data- Attack Statistics*. Retrieved from InfoSec Institute: <http://resources.infosecinstitute.com/category/enterprise/phishing/the-phishing-landscape/phishing-data-attack-statistics/#gref>
- Davis. (2017). *Types of Samples*. Retrieved from [ucdavis: http://psc.dss.ucdavis.edu/sommerb/sommerdemo/sampling/types.htm](http://psc.dss.ucdavis.edu/sommerb/sommerdemo/sampling/types.htm)
- Dennis, A., & Tegarden, B. H. (2005). *Systems Analysis and Design with UML Version 2.0*. John Wiley & Sons, Inc.

- Desai, Mukti, & Giyanani. (2014). Spam Detection using Natural Language Processing. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 116-119.
- Dinaburg. (2016). *Bitsquatting: DNS Hijacking without exploitation*. Retrieved from dinaburg.org: <http://dinaburg.org/bitsquatting.html>
- El-Din, & Salah. (2005). To Deceive or Not to Deceive! Ethical Questions in Phishing Research. *To Deceive or Not to Deceive! Ethical Questions in Phishing Research*.
- Fadi. (2006). *Chapter 3 Research Methodology - UM Repository*. Retrieved from UM Repository: repository.um.edu.my/76/4/CH3.pdf
- Guardian. (2017, January 27). *How phishing emails target banks accounts, e-payment channels*. Retrieved from The Guardian: <https://guardian.ng/technology/how-phishing-emails-target-banks-accounts-e-payment-channels/>
- Gudkova, D., Maria, V., Nadezhda, D., & Tatyana, S. (2016, August). *Kaspersky Lab*. Retrieved from [kasperskycontenthub.com: https://kasperskycontenthub.com/securelist/files/2016/08/Spam-report_Q2-2016_final_ENG.pdf](https://kasperskycontenthub.com/securelist/files/2016/08/Spam-report_Q2-2016_final_ENG.pdf)
- Gupta, & Shukla. (2015). System Design, Investigation and Countermeasure of Phishing Attacks using Data Mining Classification Methods and its Analysis. *System Design, Investigation and Countermeasure of Phishing Attacks using Data Mining Classification Methods and its Analysis*.
- Hautzer, Helbig, & Schiefer. (1997, June 18). *Computer Based Information and Report System for the support of Agriculture extension - Presentation of a Prototype*. Retrieved from efita: <http://www.efita.net/apps/accesbase/bindocload.asp?d=5605&t=0&identobj=hmc5LIMZ&uid=57305290&sid=57&idk=1>
- Havens. (2015, October 20). Analyzing Spear Phishing Attacks. *The PhishLabs Blog*, 1.
- Jakobsson, & Myers. (2006). Phishing and countermeasures: Understanding the increasing problem of electronic identity theft. In J. Wiley, *Phishing and Countermeasures* (pp. 1,9-12,18,20). Wiley-Interscience; 1 edition (December 15).
- Kaggle. (2015, December 30). *Bag of Words Meets Bags of Popcorn*. Retrieved from Kaggle: <https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words>
- Kiser. (2016, August 11). Algorithmia. *Introduction to Natural Language Processing (NLP) 2016*.
- Kulkarni. (2012). *Reinforcement and Systematic Machine Learning for Decision Making*. John Wiley & Sons, Inc.

- Leung. (2015, July). *Validity, reliability, and generability in qualitative research*. Retrieved from NCBI: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4535087/>
- Mackowiak. (2016). *New Report on the State of Phishing Attacks from Wombat Security Shows Significant Increases Year over Year*.
- Melvin. (2017, January 29). *Sample Bank Phishing Email*. Retrieved from Smart Green Pig: <http://smartgreenpig.secumd.org/sample-bank-phishing-email/>
- Microsoft. (2017). *Visio Professional and diagram software*. Retrieved from Microsoft: <https://products.office.com/en-ca/visio/visio-professional-business-and-diagram-software>
- Neil, & Brewerton. (2005). *Rapid Prototyping of Machine Learning Sysyems*. SAE Technical Paper.
- Okstate. (2017). *Experimental Research and Design*. Retrieved from Okstate.edu: <https://www.okstate.edu/ag/agedcm4h/academic/aged5980a/5980/newpage2.htm>
- Open Berkeley. (2016, January 6). *Phishing Example: "Paperless E2"*. Retrieved from Berkeley Information Security and Policy: <https://security.berkeley.edu/news/phishing-example-paperless-w2>
- Open Berkeley. (2017, April 24). *Berkeley Information Security and Policy*. Retrieved from UC Berkeley: <https://security.berkeley.edu/resources/phishing/phishing-examples-archive>
- Oracle. (2017, February 22). *MySQL*. Retrieved from MySQL Connector/Python Developer Guide: <https://dev.mysql.com/doc/connector-python/en/>
- Platts. (2006). Knowledge-base systems. *Rule-Based Systems*, 1-10.
- Prasad. (2016). *Top 20 Python Machine Learning Open Source Projects, updated*. Retrieved from KD nuggets: <http://www.kdnuggets.com/2016/11/top-20-python-machine-learning-open-source-updated.html>
- Python. (2017, April 4). *12.6. sqlite3 — DB-API 2.0 interface for SQLite databases*. Retrieved from Python Software Foundation: <https://docs.python.org/3/library/sqlite3.html>
- Quintero, B. (2017, June). *Advanced features & tools*. Retrieved from Virus Total: <https://www.virustotal.com/en/documentation/>
- Richert, & Coelho. (2013). *Buliding Machine Learning Systems with Python*.
- Saed. (2017). *Naive Bayesian*. Retrieved from saedsayad.com: http://www.saedsayad.com/naive_bayesian.htm

- Sagar, Naresh, & Reddy. (2013). Intelligent Phishing Website Detection and Prevention System by Using Link Guard Algorithm. *IOSR Journal of Computer Engineering (IOSR-JCE)*, PP 28-36.
- Sharma. (2015, May 28). *Supervised Learning for Text Classification*. Retrieved from The Digital Group: <http://blog.thedigitalgroup.com/rajendras/2015/05/28/supervised-learning-for-text-classification/>
- SHYNI, & EMILIN. (2014). *Detecting And Preventing Phishing Websites Dppws*. Shodhganga. India: C. EMILIN SHYNI.
- Smith. (2013, April 8). *Determining Sample Size: How to Ensure You Get the Correct Sample Size*. Retrieved from qualtrics: <https://www.qualtrics.com/blog/determining-sample-size/>
- Solutions, E. (2017, Feb 2). Why requirements are important.
- Sunil. (2015, September 13). *6 Steps to Learn Naive Bayes Algorithm (with code in Python)*. Retrieved from Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/>
- Symantec. (2016). *Internet Security Threat Report, Volume 21*. Symantec Corporation.
- Tonge, & Surbhi. (2015). Phishing Susceptibility and Anti-Phishing Security Strategies-Literature Review. *Phishing Susceptibility and Anti-Phishing Security Strategies-Literature Review*.
- UML-Diagrams. (2017). *UML Use Case Diagrams*. Retrieved from Uml-diagrams: <http://www.uml-diagrams.org/use-case-diagrams.html>
- Verizon. (2016). *2016 Data Breach Investigations Report*.
- Vie. (2000). *Understanding Data Flow Diagrams*. Retrieved from ratandon: http://ratandon.mysite.syr.edu/cis453/notes/DFD_over_Flowcharts.pdf
- Yasin, & Aduhasan. (2016). An Intelligent Classification Model For Phishing Email Detection. *International Journal of Network Security & Its Applications (IJNSA)*.

Appendices

Appendix A: Code Snippets

Appendix A.1: Sample Python Stop Words List Code

```
stop_words_list = ["a", "about", "above", "after", "again", "against", \
    "all", "am", "an", "and", "any", "are", "aren't", "as", "at", "because", "been", "before", "being", \
    "between", "both", "but", "by", "can't", "cannot", "could", "couldn't", "did", "didn't", "do", \
    "does", "doesn't", "doing", "don't", "down", "during", "each", "few", "for", "from", "further", "had", "hadn't", \
    "has", "hasn't", "have", "haven't", "having", "he", "he'd", "he'll", "he's", "her", "here's", "hers", \
    "herself", "him", "himself", "his", "how", "how's", "i", "i'd", "i'll", "i'm", "i've", "if", "in", "into", "is", \
    "isn't", "it", "it's", "its", "itself", "let's", "me", "more", "most", "mustn't", "my", "myself", "no", "nor", \
    "not", "of", "off", "on", "once", "only", "or", "other", "ought", "our", "ours", "ourselves", "out", "over", "own", \
    "same", "shan't", "she", "she'd", "she'll", "she's", "should", "shouldn't", "so", "some", "such", "than", "that", \
    "that's", "the", "their", "theirs", "them", "themselves", "then", "there", "there's", "these", "they", "they'd", \
    "they'll", "they're", "they've", "this", "those", "through", "to", "too", "under", "until", "up", "very", "was", "wasn't", \
    "we", "we'd", "we'll", "we're", "we've", "were", "weren't", "what", "what's", "when", "when's", "where", "where's", \
    "which", "while", "who", "who's", "whom", "why", "why's", "with", "won't", "would", "wouldn't", "you", "you'd", \
    "you'll", "you're", "you've", "your", "yours", "yourself", "yourselves"]
```

Appendix A.2: Sample Python Remove Stop Words Code

```
def remove(message):
    message_words = tokenizer(message.lower())
    filtered = list(set(message_words).difference(set(stop_words_list)))
    return filtered
```

Appendix A.3: Training Dataset of Phishing Key Words in relation to Banking

```
c = objclassify()
c.train(['now','click','below','here','credentials','exe','vbscript','pdf','details','order','docs','doc','attention','alert',
'pin','code','number','credit','card','open','spam','visit','sign','free','file','confirm','account','password','verify','download',
'preview','send','link','reset','attachment','attached','order','invoice','agent','swift','payments','purchase','disabled',
'credentials','check','access','quote','request','office','google','shared','drive','request','details','deactivation','dropbox',
'transactions','funds','claims','hesitate','urgent','attention','warning','proforma','view','expire','login','update','review',
'id','passcode','redirected','clicking','expire','immediately'], 'phish')
```

Appendix A.4: Scan Link in using Virus Total API

```
def validate(link):
    print "[+] Unpacking phishing Link : " + link + "\n"
    r1 = urlparse.urlsplit(link)
    print r1
    r2=r1.geturl()
    print "===== " + "\n"
    print "Phishing URL is : " + r2
    print "===== " + "\n"
    NetLoc=r1.netloc
    print "Network location part is : " + NetLoc
    vt = virustotal2.VirusTotal2('a5d6ce5219e5e6135d390ac292f01ac20c214b9f00667792ddd50af467cc089e')
    if NetLoc and not NetLoc.isspace():
        try:
            # the NetLoc is non-empty
            NetLocAsString = str(NetLoc)
            ip_report = vt.retrieve(NetLoc)
            total_pos = sum([u["positives"] for u in ip_report.detected_urls])
            total_scan = sum([u["total"] for u in ip_report.detected_urls])
            count = len(ip_report.detected_urls)
            if int(total_pos/count) >= 1:
                outcome = "malicious"
                MaliciousLink = link
                URL = NetLocAsString
                MaliciousLinkIP = str(get_ips(link))
                MaliciousLinkIPDetails = str(get_ips_for_host(link))
                return outcome , MaliciousLink, URL, MaliciousLinkIP, MaliciousLinkIPDetails
            elif int(total_pos/count) < 1:
                outcome = "safe"
```

Appendix A.5: Acquiring Email Attachment Name for Extension Analysis

```
def getAttachmentName(id):
    msg = Message.message_from_string(body)
    MessageToString = str(msg)
    print "[+] Acquiring attachment name...." + "\n"
    x = re.findall(r'Content-Disposition: attachment; filename="(.*?)"',MessageToString,re.DOTALL)
    filename = " ".join(x)
    print "[+] Acquiring attachment name is : " + filename + "\n"
    return filename
```

Appendix B: Turnitin Report


FINALIST THESES/DISSERTATION .. Masters Theses - 2017 Final Submission:

Originality

GradeMark

PeerMark

An adaptive Phishing Email Detection Algorithm
BY EDWIN OROKO ORINA

turnitin 

15%
SIMILAR



**An Adaptive Phishing Email Detection Algorithm: Case of Phishing Attacks
in the Banking Industry**

By
Oroko Edwin Orina

A Dissertation submitted in partial fulfilment of the requirement for the award of a Master
of Science Degree in Information System Security (MSc. ISS) at Strathmore University.

Faculty of Information Technology

No Service Currently Active



PAGE: 1 OF 81

