Addis Ababa University

College of Natural Sciences

Nature Inspired Algorithms along with Support Vector Machine for Detecting Web Phishing Attack

Fitsum Alemu Degfu

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

June 2018

Addis Ababa University

College of Natural Sciences

Fitsum Alemu Degfu

Advisor: Mulugeta Libsie (PhD)

This is to certify that the thesis prepared by Fitsum Alemu Degfu, titled: *Nature Inspired Algorithms along with Support Vector Machine for Detecting Web Phishing Attack* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	Signature	Date
Advisor:			
Examiner:			
Examiner:			

Abstract

The advancement of Computer Technology has grasped attention of individuals living on the 21th century. This advancement has made life much simpler where one can use phones in his/her hand to access different kinds of facilities without even making a few yards of movement.

Web Technology is one example that has materialized to this generation due to the invention of the Internet. The Technology allows Internet users to access different kinds of services in a modern and easy way. A user only needs a web browser software or an application that allows him/her to perform bank transaction without being physically present on the bank compound.

Even though efforts made to enhance the technology, there are still tradeoffs on keeping user's personal information private and in a secured manner. This has been a major problem since the innovation of the Web.

There are different kinds of security threats on Web Technology mainly; Web based Threats, Denial of Service Attack, and Phishing.

Phishing is a web based attack that works by creating a fake version of the real sites web interface to gain the users trust. It is a combination of social engineering and technical exploits designed to convince a victim to provide personal information, usually for monetary gain of the attacker. Phishing has become the most popular practice among the crimes performed on the web especially on e-commerce services and social media.

Besides, the attack is becoming more frequent and sophisticated which makes it difficult to tackle. This is causing a significant impact on both service provides and consumers since it involves the risk of identity theft and financial losses that leads to loss of trust by service customers.

In this thesis we have proposed a novel classification approach which uses Support Vector Machine (SVM) optimized with Firefly Algorithm (FA) to detect phishing attack. The parameters of SVM are optimized with the help of FA for detecting phishing attack by a specific website. We implemented a prototype web browser which can be used as an agent for the purpose of retrieving Web URL and reporting phishing attack to user. We have found an acceptable range of accuracy and minimized error rate on the classification. The user testing also shows a convenient way of reporting the attack.

Keywords: Web, Website, Phishing, phishy, Phishtank, Nature Inspired Algorithm, Firefly Algorithm, Support Vector Machine

Acknowledgments

First and for most I thank God for the life and achievements I had so far and for what is to come, it has been nothing but a bliss. I would also like to express my deepest gratitude to my Advisor Dr. Mulugeta Libsie for his supreme encouragement, guidance, and support.

Secondly, I would like to thank Research Coordinating Committee for providing thesis guideline and templets for the documentation.

Thirdly my colleagues, especially Ms. Sofanit Wubeshet, Mr. Tewodros Wondifraw, Mr. Nebiat Fikru, Mr. Natnael Argaw, and others share a great deal of my gratitude for their continuous provision and support.

Last but not list, I would like to thank those individuals who have been a volunteer in the testing process of specific components. I would like also to expand my deepest gratitude for all individuals who have made significant contribution to the success of this thesis.

Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms and Abbreviations

APWG Anti Phishing Working Group

ACO Ant Colony Optimization

BA Bees Algorithm

BFOA Bacterial Foraging Optimization Algorithm

CCH Context Histogram

EA Evolutionary Algorithms

E-fraud Electronic fraud

GA Genetic Algorithm

MCAR Multi-class Classification based on Association Rule

NIT Nature Inspired Techniques

PSO Particle Swarm Optimization

SVM Support Vector Machine

TF-IDF Term-Frequency-Inverse-Document-Frequency

TTL Time to Leave

Chapter 1 : Introduction

1.1 BACKGROUND

Electronic fraud (or e-fraud for short) is a fraudulent action that is perpetrated using electronic technologies and devices, such as computers, Internet, email, telephone networks, and mobile phones [1]. There are three main types of electronic fraud which are most prevalent today [1]: Email Spam, Phishing and Network Intrusion. Different branches and techniques of artificial intelligence and statistics are being used in e-fraud detection systems, among which is nature-inspired techniques [1]. Nature inspired techniques, as the name implies, are artificial intelligence techniques which are inspired by the way natural systems work. For instance, the Ant Colony Optimization (ACO) is based on the way ants find the shortest path between their nest and the food source [2], or evolutionary algorithms (EAs) are population-based metaheuristic optimization algorithms that are inspired by biological evolution: reproduction, mutation, recombination, and selection [3]. Nature-inspired techniques are very effective in solving classification and optimization problems [1]. The first reason for this success is their versatility where they don't require to have a complete knowledge of the domain they are used for and no need of tuning [4]. In many cases of e-fraud detection problems, we do not have complete information of the environment which does not cause any difficulty for nature-inspired techniques. In addition, these techniques are very robust where they can adapt as the environment changes [5]. This is particularly very essential in the dynamic world of e-fraud detection.

Support Vector Machine (SVM) is a linear machine learning method (MLM) based on structural risk minimization of statistical learning theory (STL) [6]. SVM has various attributes that make it suitable for implementing an efficient e-fraud detection system. The first attribute is that it shows good generalization skill, without the need of any prior knowledge. Furthermore, it does not suffer from the local minimum and it can handle noisy datasets.

One of the main types of e-fraud is Phishing (using fake websites) which attract people to visit fraudulent websites and making them to enter confidential data like credit card number, username and password. In the computer field, stealing information is a simple trick in which a hacker creates the duplicate page of a site and asks individuals to enter their details or credentials [7]. When a user enters his/her credentials such as username, password or credit card number the whole data goes to the hacker which s/he later uses. This type of act is an illegal process of getting some one's

information but nowadays it is a major problem in social networking sites and net banking. There are many common ways of identifying phishing websites [8]: checking the URL, using well known anti-phishing software tools, using browsers capacity of identifying phishing, using fake account information for trial, using well known search engines, etc.

Different attempts have been made to deal with the growing number of reported phishing incidents, including legislation, user training, public awareness, and technical security measures. The first technique is browser based where the browser shows a warning to the user when it suspects an attack [8]. The browser also checks the security certificate of the website and reports if it finds anything suspicious. The main problem with browser warnings is that research shows that users do not pay attention to warnings and messages and press the OK and accept buttons automatically. There are many researchers conducted on studies of security toolbars and other browser security indicators for the purpose of identifying and protecting from phishing and found them all ineffective at preventing phishing attacks.

Hence, this research is aimed at creating a resilient and effective method that uses nature inspired algorithms and support vector machine for detecting web phishing attack.

1.2 MOTIVATION

There has been reports what most Internet users heard of regrading vulnerabilities on web. Phishing is one of the continual threat, and the risk is even larger in social media such as Facebook, Twitter, and Google+. There can be several reasons to be mentioned for the initiation of this thesis.

To start with, the major problems that we have been facing throughout the years happens when we try to access different resources in the web but we end up in loosing our private information by following links provided in webpages and through our email.

The other issue is the very nature of this attacks changing in the course of time made us to have a great desire to come up with a robust solution that can cope up with this attacks.

Last but not list, recent advancement in nature inspired algorithms that emulates natural behavior of different living things and using them for the purpose of finding optimal solutions from a large set of local solutions is what motivates us to start this thesis. The success they have in different areas especially the in tackling optimization problems that has been used in most of security bridges like intrusion detection have a very impact on the initiation of this research.

1.3 STATEMENT OF THE PROBLEM

Phishing in the first place can be taken as a criminal activity which leads the individuals who perform such activity to legal punishment [9]. There are various challenges that phishing presents:

The first challenge that phishing presents is loss of trust by customers on using a system at all because of the vulnerabilities they will be exposed in terms of losing sensitive information like credential information and credit card information. Loss of trust makes a company to loss a huge amount of customers which affect in two ways: the first is the number of system users will be reduced so that the organization will be forced to return back to the old way of working style and mechanism to accomplish work. The other is Montreal loss due to loss of creational information

The second challenge that phishing presents is for financial institutions that involve in online transactions which is the major problem nowadays that leads users to lose a huge amount of money due to loss of credit card information. This leads financial institutions to loss a huge amount of customer and decreases the profit of the institution.

The other challenge phishing presents is to the larger social medias such as Facebook, Twitter, and Google+ especially in the loss of privacy due to loss of credential which is critical aspect of this systems that should be satisfied.

1.4 OBJECTIVES

General Objective

The objective of this research is to use Nature Inspired Algorithms for parameter optimization and Support Vector Machine for classification of websites as phisher or actual to detect the phishing attack and report to the user**.**

Specific Objectives

For achieving the general objective, the following specific objectives are identified:

- Making a detailed analysis and study on Nature Inspired Algorithms and techniques for the purpose of e-fraud.
- Generating corpus data for testing how this algorithm works and measure the effectiveness.
- Compare and contrast in between various Nature Inspired Algorithms for helping further study.

- Create a general model for presenting how SVM works with optimized parameter values by Nature Inspired Algorithms.
- Develop a prototype for testing how Nature Inspired Algorithms perform the task of identifying phishing attack alongside with Support Vector Machine.

1.5 METHODS

The methods that will be used in order to achieve the objectives are: literature review, detailed study of existing model and prototyping.

Literature Review

Related literatures on E-fraud detection using SVM optimized with Nature Inspired Algorithms from different sources such as published papers written by a variety of researchers and other materials will be reviewed in depth to obtain understanding of the area in detail.

Detailed Study of Existing Models

Before designing a model for this new approach there will be a detailed analysis on existing model used by different researchers reviewed from different articles in different area and using different approaches will be performed.

Prototyping

In order to test the proposed model, a prototype will be developed. The prototype will encompass the entire solution according to the procedures and guidelines for implementing the prototype.

1.6 SCOPE AND LIMITATIONS

The scope of this research is limited to: -

- Feature selection for website phishing
- Classify websites as actual and phisher
- Test how Nature Inspired Algorithms can work with SVM for the purpose of phishing detection
- Generate informative warning to users about the phishing attack

1.7 Application of Results

The completion of this research will add value to the field of computer security in fostering the available Nature Inspired Algorithms to be used for different kinds of security issues which are serious problems nowadays.

Most companies that involve in online transaction especially financial institutions will be largely benefited from the accomplishment of this research by protecting both internal and external users of the system from phisher attack through reporting and providing informative warning about the attack.

Besides finical institutions, it also provides a protection frame work for social media services in preventing users from losing credential. This has a great impact on building confidence on customers on using these services and protect privacy of their information.

1.8 Organization of the Thesis

The rest of this thesis report is organized as follows. Chapter Two introduces an overview of web security threats and nature inspired algorithms (NIA) in different area of researches. Chapter Three presents reviews of related researches conducted on detection of phishing attack with various techniques. Chapter Four describes the general design and architectural model for preventing a user from phishing attack usin*g* SVM optimized with NIA with Rule based addons. Chapter Five discusses the developed prototype and its evaluation. Chapter Six presents conclusions, contribution and future works of the thesis.

Chapter 2 : Literature Review

2.1 THE WEB

In the early days of the Internet, the World Wide Web consisted only of Websites. These were essentially information repositories containing static documents. Web browsers were invented as a means of retrieving and displaying those documents. The flow of information was one-way, from server to browser. Most sites did not authenticate users, because there was no need to. Each user was treated in the same way and was presented with the same information [10].

Today, the World Wide Web is almost unrecognizable from its earlier form. The majority of sites on the web are in fact applications. They are highly functional and rely on two-way flow of information between the server and browser. They support registration and login, financial transactions, search, and the authoring of content by users. The content presented to users is generated dynamically on the fly and is often tailored to each specific user. Much of the information processed is private and highly sensitive. Security, therefore, is a big issue. No one wants to use a web application if s/he believes her/his information will be disclosed to unauthorized parties.

Web applications bring with them new and significant security threats. Each application is different and may contain unique vulnerabilities. Most applications are developed in-house many by developers who have only a partial understanding of the security problems that may arise in the code they are producing. To deliver their core functionality, web applications normally require connectivity to internal computer systems that contain highly sensitive data and that can perform powerful business functions. In the earlier days if one wanted to make a funds transfer, one should visited a bank, and the teller performed the transfer but today, one can visit a web application and perform the transfer. An attacker who compromises a web application may be able to steal personal information, carry out financial fraud, and perform malicious actions against other users.

Here are some web application functions that have risen to prominence in recent years [10]: Shopping (Amazon), Social Networking (Facebook), Banking (Citibank), Web search (Google) Auctions (eBay), Gambling (Betfair), Web logs (Blogger), Web mail (Gmail), Interactive information (Wikipedia). Applications that are accessed using a computer browser increasingly overlap with mobile applications that are accessed using a smartphone or tablet. Most mobile applications employ either a browser or a customized client that uses HTTP-based API to

communicate with the server. Application functions and data typically are shared between the various interfaces that an application exposes to different user platforms. In addition to the public Internet, web applications have been widely adopted inside organizations to support key business functions. Many of these provide access to highly sensitive data and functionality: Human Resource applications allowing users to access payroll information, give and receive performance feedback, and manage recruitment and disciplinary procedures. Administrative interfaces to key infrastructure such as web and mail servers, user workstations, and virtual machine administration. Collaboration software are used for sharing documents, managing work flow and projects, and tracking issues. These types of functionality often involve critical security and governance issues, and organizations often rely completely on the controls built into their web applications. Business applications such as enterprise resource planning (ERP) software, which previously were accessed using a proprietary thick-client application, can now be accessed using a web browser.

There is no doubt that web application security is a current and newsworthy subject. For all concerned, the stakes are high: for businesses that derive increasing revenue from Internet commerce, for users who trust web applications with sensitive information, and for criminals who can make big money by stealing payment details or compromising bank accounts. Reputation plays a critical role. Few people want to do business with an insecure website, so few organizations want to disclose details about their own security vulnerabilities or breaches. Hence, it is not a trivial task to obtain reliable information about the state of web application security today.

2.2 WEB SECURITY THREATS

The security of Web applications has become increasingly important in the last decade. With more and more Web-based applications that deal with sensitive financial and medical data, it is crucial to protect these applications from different kinds of attack. A security assessment by the Application Defense Center, which included more than 250 Web applications from e-commerce, online banking, enterprise collaboration, and supply chain management sites concluded that at least 92% of Web applications are vulnerable to some form of attack [11]. Another survey found that about 75% of all attacks against Web servers target Web-based applications [12].

Motivated by the lure of profits from the sale of stolen confidential information, cyber criminals today are shifting to the Web as their target area, which provides an ideal environment for cybercrime.

Web threats pose a broad range of risks, including financial damages, identity theft, loss of confidential business information, theft of network resources, damaged brand or personal reputation, and erosion of consumer confidence in e-commerce. These high stakes, the pervasive use of the Web, and the complexity of protecting against Web threats combine to form perhaps the greatest challenge to protecting personal and business information in a decade. Some of the threats are shown in Figure 2.1 [13] and they are briefly described in order.

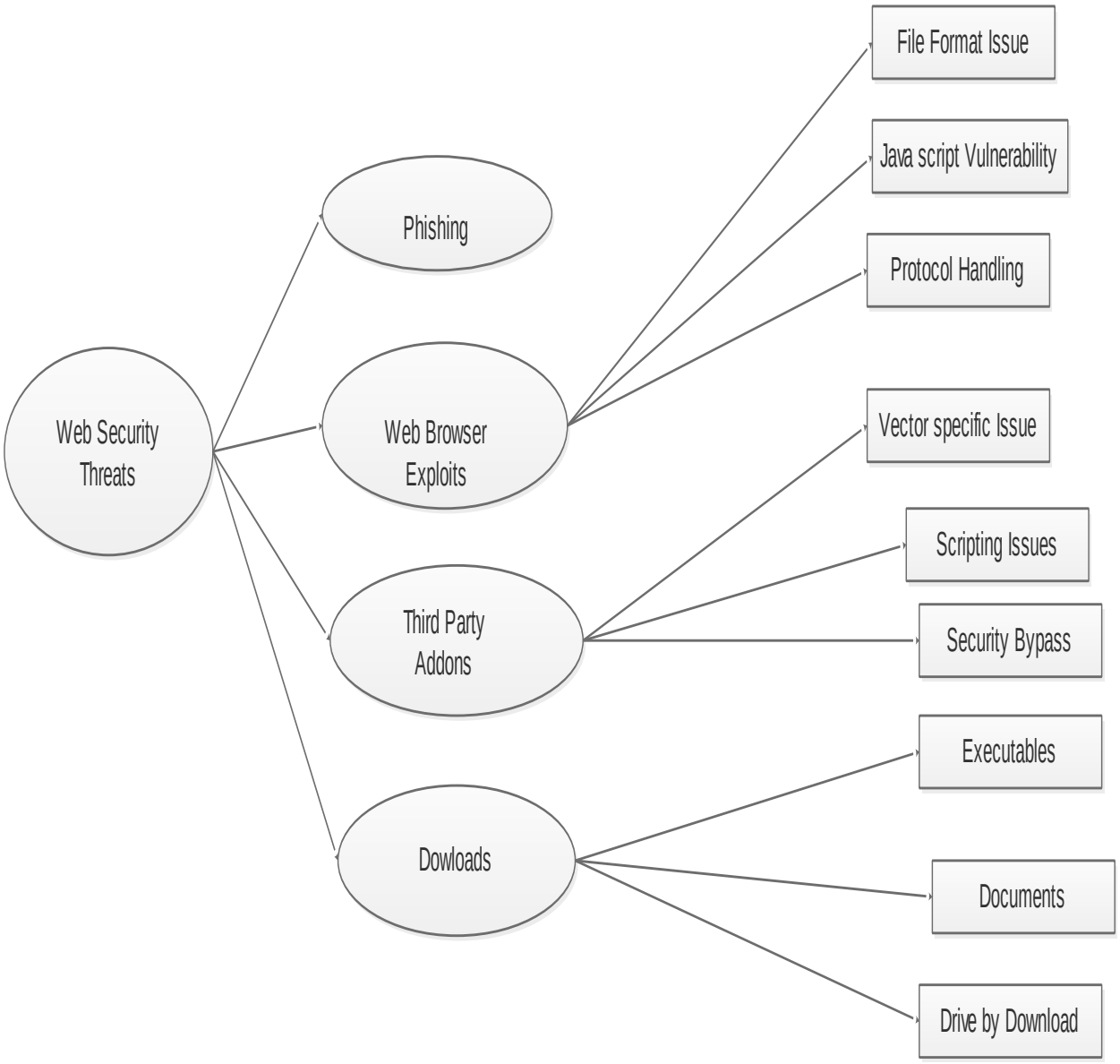

Figure 2.1: Web Security Threats

a. Phishing

One of the less advanced, but nonetheless effective threats is phishing. The term refers to attacks where the victim is led to believe that he or she is on a legitimate website, when in fact it is just a copy of the real one [14]. This attack relies on the fact that anyone can create their own website and any website can look like any other. A real-world example would be a fake ATM that is put in the middle of a busy shopping center. There would be very few signs to show victims that it is not a real ATM until no money comes out. Similarly, in a phishing attack victims may think that they are on their bank's website, and therefore do not think twice about using pin numbers as requested. This attack is not limited to banking systems. Phishing attacks have been known to target company email websites (webmail), and popular sites like Amazon or eBay. Users can identify a phishing website in a number of ways. The first is to look at the URL [15]. Another effective prevention technique is to never follow links by email but to type them in or use bookmarks [16, 17, 18]. Although not foolproof, these methods make it harder for attackers to push forward.

b.Web Browser Exploits

Cybercriminals have also setup websites that exploit security holes in the web browser. This technique allows them to gain access without the victim's knowledge. Web browsers are complex software [19]. They have to handle various file formats, such as images, sound and HTML, JavaScript, and a multitude of other technologies. All of these features add to the attack surface of the web browser, thus making the technology relatively weak from a security standpoint. Yet this same functionality is what makes it so useful. Not surprisingly, both Internet Explorer and Firefox have had their fair share of security vulnerabilities. In fact, as of 2009 [20], Secunia estimates that Internet Explorer 7 had over 90 vulnerabilities, while Firefox 3.0 had over 130 vulnerabilities. Some of these security flaws could be exploited to allow individuals to remotely get access to private emails, sensitive documents and anything else that the user running the web browser has access to.

c. Third party add-ons

The majority of websites require the use of third party add-ons [21] such as Adobe Flash player and Acrobat Reader. Both of these widely-used products have become a favorite target for cybercriminals. As more administrators and home users update their machines with the latest security updates and patches for their browsers, as well as the ability to automate the process, it becomes harder to use web browsers as an attack vector. However, although they may be updating their browser software, it is also true that many people forget to update third party add-ons like Flash

player. In 2009 [22] a number of malware "in the wild" (out there on the Internet) have exploited the PDF file format, Adobe Acrobat, Flash, a number of ActiveX components and Java. These third-party add-ons are used to push users to other websites that have been compromised.

d. Downloads

While automating remote code, execution is very attractive for attackers, there are many times when this level of sophistication is not required to compromise end user computers [21]. In fact, some attacks still rely on end users downloading executable files. To aid attackers, legitimate websites, some of which are high profile, are being compromised. As soon as these sites become infected, they can start serving malware, thus exploiting users tendency to trust content based on reputation. When a legitimate news site asks end users to download an executable file (e.g., codec) to view an intriguing video, many will comply because they trust that website. Malware creators use a variety of techniques to convince users to visit poisoned sites, search results and download executables. The bad guys love to play on two characteristics of human nature: fear and curiosity. In 2009, the rise of 'scareware' has been considerable [23]. Playing on people's fears that their machine has been infected with malware, users are encouraged to download antivirus software. This is nothing but malware that infects the machine and demands payment if the user wants to uninstall the software. Another threat is the use of poisoned search engine results. The bad guys create numerous websites for well-known keywords and use these sites to feature high upon on web searches. When a user searches for a particular name or subject and clicks on a poisoned link, s/he is taken to a fake website where they are told to either download software to continue or else malware is downloaded while they are looking at the content.

2.3 What Phishing has Brought

Despite growing efforts to educate users and create better detection tools, users are still very susceptible to phishing attack. Unfortunately, due to the nature of the attack, it is very difficult to estimate the number of individuals who actually fall a victim. A 2006 report by Gartner [24] estimated the costs at $1,244 per victim, an increase over the $257 they cited in a 2004 report. In 2007 Moore and Clayton [25] estimated the number of phishing victims by examining web server logs. They estimated that 311,449 people fall for phishing scams annually, costing around 350 million dollars. Another study in 2007 by Florencio and Herley [26] estimated that roughly 0.4% of the population falls for phishing attacks annually. Phishing works because users are willing to trust

websites that appear to be designed well. In a 2001 study on website credibility, Fogg *et al.* [27] found that the "look and feel" of a website is often most important for gaining users trust.

A 2006 phishing study by Dhamija *et al.* [8] found that 90% of the participants were fooled by phishing websites. The researchers concluded that current security indicators (i.e., the lock icon, status bar, and address bar) are ineffective because 23% of the participants failed to notice them or because they did not understand what they meant.

In a similar study, Downs *et al.* [28] showed participants eight emails, three of which were phishing. They found that the number of participants who expressed suspicion varied for each email; 47% expressed suspicion over a phishing message from Amazon, whereas 74% expressed suspicion over a phishing message from Citibank. Those who had interacted with certain companies in the past were significantly more likely to fall for phishing messages claiming to be from these companies. Participants were also likely to ignore or misunderstand web browser security clues. As shown in Figure 2.2 [29], online industries are mostly targeted by phishing attacks. During the fourth quarter of 2016, 19.6 percent of phishing attacks worldwide were directed towards financial institutions. Payment services accounted for 11.33 percent of phishing attacks.

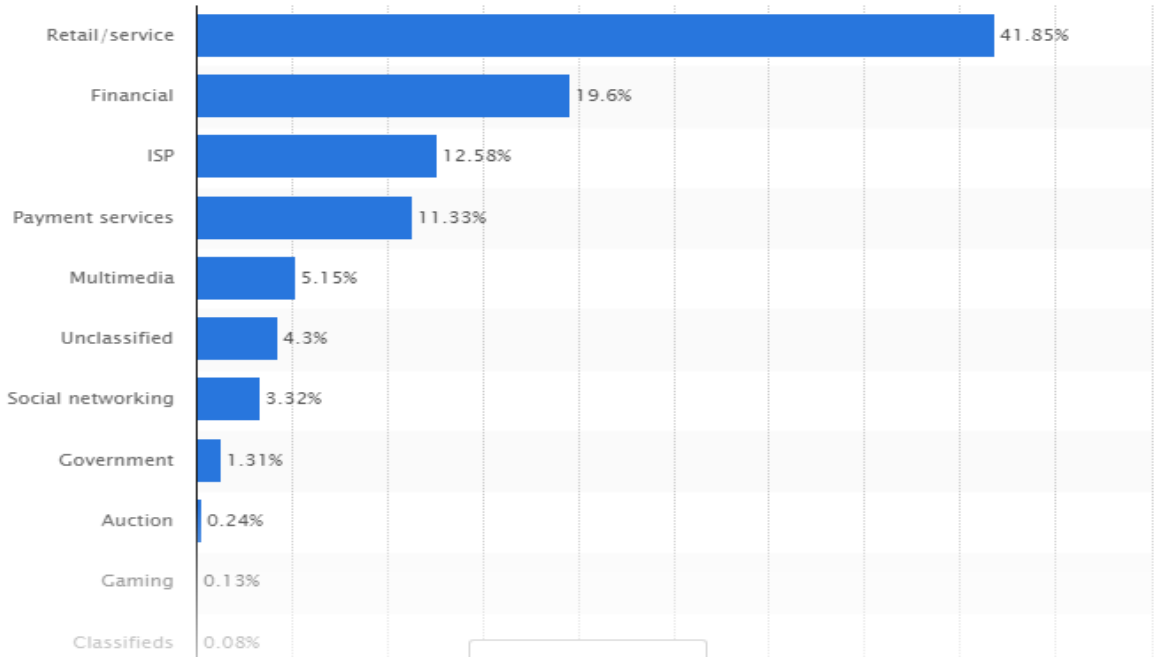

Figure 2.2: Different Industries Affected by Phishing in 2016

2.4 How to Deal with Phishing

In order to be protected from any e-fraud we emphasis on the two pillars of protection mechanism which are prevention and detection.

2.4.1 Prevention and Reporting Checklist for Phishing Schemes

One of the most basic measures that government and private sector are taking to protect the public from phishing is the provision of specific advice to the public about how to avoid phishing schemes and how to prevent and report phishing schemes. The following list of advice to the public are derived from information provided by the APWG [30]:

a)Prevention: What to Do

- Protect computer with anti-virus software, spyware filters, e-mail filters, and firewall programs, and make sure that they are regularly updated.
- Consider installing a Web browser tool bar to be protected from known phishing fraud websites. (Check the browser or e-mail provider for such toolbars.)
- Ensuring that Internet browser currently used is up to date and security patches applied.
- In particular, people who use the Microsoft Internet Explorer browser should immediately go to the Microsoft Security home page http://www.microsoft.com/security/ to download a special patch relating to certain phishing schemes.
- Be suspicious of any e-mail with urgent requests for personal financial information or threats of termination of online accounts.
    - Unless the e-mail is digitally signed, there is no means of being sure it wasn't forged or spoofed.
    - Phishers typically ask for information such as username, password, credit card number, social security number, etc.
    - Phisher e-mails are typically not personalized, while valid messages from bank or e-commerce Company generally are.
- When contacting to financial institution, use only channels that are known from independent sources that are reliable (e.g., information on bank card, hard-copy correspondence, or monthly account statement), and don’t rely on links contained in e-mails, even if the web address appears to be correct.
- Always ensure using a secure website when submitting credit card or other sensitive information via Web browser.

- To make sure that we are on a secure Web server, check the beginning of the Web address in browsers address bar - it should be "https://" rather than just "http://."
- Regularly log into personal online accounts.
- Don't leave them for as long as a month before you check each account.
- Regularly check your bank, credit and debit card statements to ensure that all transactions are legitimate.
- If anything is suspicious, contact the bank and all card issuers.

b)Prevention: What Not to Do

- Don't assume that it is easy to correctly identify a website as legitimate just by looking at its general appearance.
- Don’t use the links in an e-mail to get to any webpage, if there is a suspect that a message might not be authentic.
- Instead, call the company on the telephone, or log onto the website directly by typing in the Web address in a browser.
- Avoid filling out forms in e-mail messages or pop-up windows that ask for personal financial information.
- Communicate information such as credit card numbers or account information via a secure website or the telephone.

c)Reporting: Suspicious E-mails and Websites

- Always report a "phishing" or “spoofed” e-mail or website to the following groups, whether or not one responded to that phishing e-mail or website:
    - Forward the e-mail to reportphishing@antiphishing.com
    - Forward the e-mail to the "abuse" e-mail address at the company that is being spoofed (e.g. "spoof@ebay.com")

2.4.2 Detection for Phishing Schemes

Current phishing detection approaches fall into three main categories [31]: (1) Non-content-based approaches that do not use content of the site to classify it as authentic or phishing, (2) Content based approaches that use site contents to catch phishing, and (3) Visual similarity based approaches that identify phishing using their visual similarity with known sites. Other anti-phishing approaches

include detecting phishing emails (rather than sites) and educating users about phishing attacks and human detection methods.

a) Non-content-based approaches

Non-content-based approaches include URL and host information based classification of phishing sites, blacklisting and whitelisting methods.

In URL based schemes, URLs are classified based on both lexical and host features. Lexical features describe lexical patterns of malicious URLs. These include features such as length of the URL, the number of dots, special characters it contains. Host features of the URL include properties of IP address, the owner of the site, DNS properties such as TTL, and geographical location.

In blacklisting approaches, users report or companies seek and detect phishing sites' URLs which are stored in a database.

Whitelisting approaches seek to detect known good sites, but a user must remember to check the interface every time s/he visits any site.

b) **Content based approaches**

In content-based approach, phishing attacks are detected by examining site contents. Features used in this approach include spelling errors, source of the images, links, password fields, embedded links, etc. along with URL and host-based features.

c) Visual similarity-based phishing detection

This approach uses screenshot of webpages to detect phishing sites. Besides it uses Contrast Context Histogram (CCH) to describe the images of webpages and k-mean algorithm to cluster nearest key points. Finally, Euclidean distance between two descriptors is used to find matching between two sites.

2.4.3 TECHNIQUES THAT USE THE APPROACHES ON DETECTION

There are mainly two techniques that can be used to detect web phishing: browser based and data based identification and detection but as we know browser based detection is not as effective still now.

Data based identification is used in many kinds of e-fraud detection. Some of the data based identification techniques are:

- Nature Inspired techniques
- Machine learning techniques (support vector machine) and
- Neural Networks

2.5 NATURE INSPIRED TECHNIQUES

Nature inspired computing draws on the principles of emergence, self-organization and complex systems [32, 33]. It aims to develop new techniques, algorithms and computational applications by getting ideas from observing how nature behaves to solve complex problems. Taking animals for example, evolutionary pressure forces them to develop highly optimized organs and skills to take advantages of fighting for food, territories and mates. Some of the organs and skills can be well refined as optimization algorithms, and the evolution is a process to fine-tune the parameter settings in the algorithms.

2.5.1 Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic approach to solve problems that has been inspired by the ants' social behaviors in finding shortest paths [34]. Real ants walk randomly until they find food and return to their nest while depositing pheromone on the ground in order to mark their preferred path to attract other ants to follow. If other ants travel along the path and find food, they will deposit more pheromone to reinforce the path for more ants to follow [35]. In the past decades, substantial amount of research has been done to both develop the ACO algorithm itself and practical applications to solve relevant problems in the real world. The initial ACO algorithm, Ant System (AS), was proposed by Marco Dorigo in 1992 in a PhD dissertation [36].

The ACO was initially introduced as a cooperative learning approach to the Travelling Salesman Problem, and it now has been applied to a wide range of combinatorial optimization problems, including scheduling (e.g., Job-shop scheduling problem [37], Single machine total tardiness problem [38]), vehicle routing (e.g., Capacitated vehicle routing problem [39], Stochastic vehicle routing problem [40]) and assignment problems (e.g., Frequency assignment problem [41]). This algorithm can be applied in the bionic management study such as the optimization of the distribution and location of a new town during the integration of rural with a city.

2.5.2 Bees Algorithm (BA)

The BA has been inspired from the food foraging behavior of honey bees whose colony can typically extend itself over 10 km to 14 km and in multiple directions simultaneously to exploit a large number of food sources [42]. The colony tends to attain the optimal use of its members. Theoretically, the richer (more nectar or pollen) and closer a food source is, the more bees that would be sent to it [43]. The food foraging process begins with scout bees searching activities. They

fly randomly to explore all the food sources in all directions. When they return to the hive, they report the searching results to the others by performing waggle dance [44] which communicates these important pieces of information about the food source, including its direction, distance and quality rating. The waggle dance is essential in the food source evaluation and helps the colony to deploy flower bees to the food sources precisely. The BA is a population-based optimization algorithm first developed in 2005. The original pseudo code for the algorithm is shown in Algorithm 2.1 [45]:

Input Instance x e I of II_{opt}

Set algorithm parameters ()

Initialize Popo ←Initial Population with random solutions.

Evaluate fitness of the population (Popo).

While (stopping criterion not met) **do**

//Forming new population.

Select sites for neighborhood search.

N(s)

Recruit bees for selected sites (more bees for best e sites) and evaluate fitness.

Select the fittest bee from each patch.

Assign remaining bees to search randomly and evaluate their fitness.

End While.

Generate a new population of scout bees

S_{best}←Optimal solution Pop

Output: S_{best} candidate to be the best-found solution x e I

Algorithm 2.1: The Original Algorithm for BA

BA has been applied to many combinatorial optimization problems, ranging from manufacturing [46] to computer image analysis [47], and it is proved to have a very robust performance and a high success rate compared with other intelligent optimization methods. BA can also be applied in the research of face detection and 3D modelling of bio-robot design and development.

2.5.3 Genetic Algorithm (GA)

GA generates solutions to search, optimization and machine learning problems via applying techniques inspired by biological evolution. Genetic algorithms were initially developed in computer

simulations in the early 1950s [48]. Later on, computer simulation of evolution became very popular, and Fraser. Burnell [49] published a book to systematically describe the simulations which contains all the basic components of modern genetic algorithms. GA adopts some genetic terminologies, including:

1. Chromosome is an encoding of a solution to an optimization problem. The solutions are typically represented in binary [50].
2. Selection is a stage in the GA where individual genomes are chosen for breeding new generation.
3. Crossover and Mutation both are genetic operators applied to a pair of parents when they reproduce to alter their genetic composition [51].

To perform a GA, the first step is to initialize the population that normally is composed of randomly generated individuals covering the entire range of possible solutions, and size of the population is determined by the nature of the problem itself. The next step is to evaluate the fitness of each member of the population. A fitness function is employed in this stage to provide fitness values for each individual, and the results are then normalized in order to sort the entire population by descending fitness value. Once the selection process has finished, it comes to the reproduction stage where this generation performs repeatedly until one of the termination criteria is met as shown in Figure 2.3 [52]. GA has been widely adopted to solve complex problems, especially in the areas of scheduling, global optimization and control engineering [53, 54, 55].

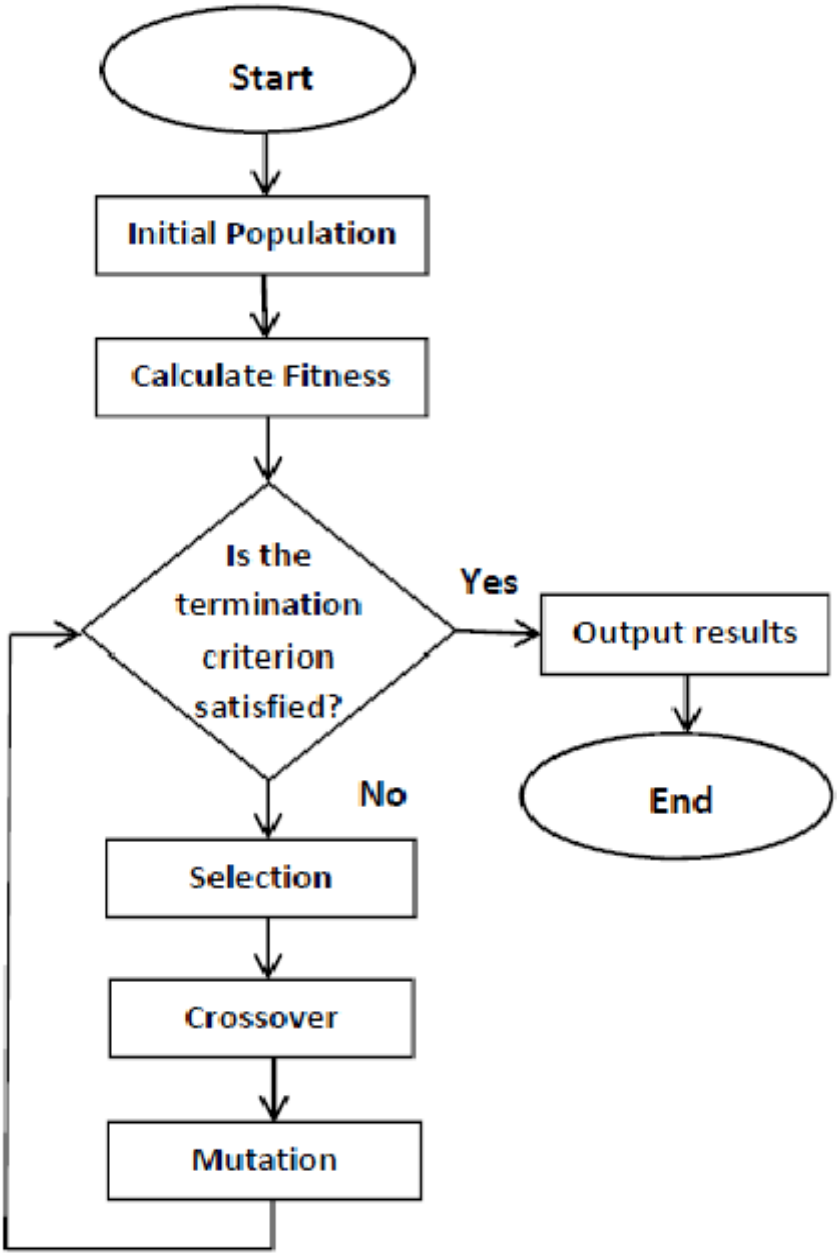

Figure 2.3: GA Stages

2.5.4 Firefly Algorithm (FA)

Fireflies produce luminescent flashes as a signal system to communicate with other fireflies, especially to prey attractions [56]. FA is inspired by the firefly's biochemical and social aspects and it is based upon the following three assumptions: (a) Each firefly attracts all the other fireflies with weaker flashes; (b) Attractiveness is proportional to their brightness which is inverse proportional to their distances; (c) No fireflies can attract the brightest firefly, and it moves randomly [57].

The brightness of firefly is associated with the objective function f(x), $x \in M_d$, assume that there exists n fireflies xi, i = 1, 2,.,n initially positioned randomly in the space. Given the distance between firefly i and firefly j as r_{ij} = d(xi, xj), the attractiveness between them should follow monotonically decreasing function, e.g.= $ke^{-r_{ij}}$, where k is the max attractiveness value. The flash intensity I is associated with the objective function f(x), i.e. I = Įf(x). Only firefly with higher flash intensity attracts the other one i.e. Ii<Ij, j = 1, 2,. n, j≠i. The algorithm is given in Algorithm 2.2 [58]. FA can be applied in the optimization of traffic system design and development of a new town.

Input: Objective function $f(x)$, $x = (x_1,\cdots,x_D)^T$

Generate initial population of fireflies xi (i =1 ,2,...,n)

Light intensity Ii at xi is determined by $f(x_i)$

Define light absorption coefficient γ

while (t<MaxGeneration)

for i =1:n all n fireflies

for j =1:i all n fireflies

if (Ij >I i), Move firefly i towards j in d-dimension;

end if Attractiveness varies with distance r via $\exp[-\gamma^r]$ Evaluate new solutions and update light intensity

end for j

end for i

Rank the fireflies and find the current best

end while

Post process results and visualization

Output: Best solution

Algorithm 2.2: Pseudo code for FA

2.5.5 Particle swarm optimization

The Particle Swarm Optimization (PSO) is a Swarm Intelligence method that models social behavior to guide swarms of particles towards the most promising regions of the search space [59]. It has roots in two main component methodologies. Perhaps more obvious are its ties to artificial life (A-life) in general, and to bird flocking, fish schooling, and swarming theory in particular. It is also

related, however, to evolutionary computation, and has ties to both genetic algorithms and evolution strategies [60].

Particle swarm optimization comprises a very simple concept and paradigms are implemented in a few lines of computer code. It requires only primitive mathematical operators and it is computationally inexpensive in terms of both memory requirements and speed. Early testing has found the implementation to be effective with several kinds of problems [61].

Each individual in the particle swarm is composed of three D-dimensional vectors [62], where D is the dimensionality of the search space. These are the current position xi, the previous best position pi, and the velocity vi. The current position xi can be considered as a set of coordinates describing a point in space. On each iteration of the algorithm, the current position is evaluated as a problem solution. If that position is better than any that has been found so far, then the coordinates are stored in the second vector, pi. The value of the best function result so far is stored in a variable that can be called pbest (for "previous best"), for comparison on later iterations. The objective, of course, is to keep finding better positions and updating pi and pbest. New points are chosen by adding vi coordinates to xi, and the algorithm operates by adjusting vi, which can effectively be seen as a step size. The particle swarm is more than just a collection of particles. A particle by itself has almost no power to solve any problem; progress occurs only when the particles interact.

Problem solving is a population-wide phenomenon, emerging from the individual behaviors of the particles through their interactions [62]. In any case, populations are organized according to some sort of communication structure or topology, often thought of as a social network. The topology typically consists of bidirectional edges connecting pairs of particles, so that if j is in i's neighborhood, i is also in j's. Each particle communicates with some other particles and is affected by the best point found by any member of its topological neighborhood. This is just the vector pi for that best neighbor, which we will denote as pg. The potential kinds of population "social networks" are hugely varied, but in practice certain types have been used more frequently. In the particle swarm optimization process, the velocity of each particle is iteratively adjusted so that the particle stochastically oscillates around pi and pg locations. The (original) process for implementing PSO is shown in Algorithm 2.3

Input: group of random particles

Output: Best Solution (pBest)

```
For each particle
   Initialize particle
END
Do
   For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pBest) in history
         set current value as the new pBest
   End
   Choose the particle with the best fitness value of all the particles as the gBest
   For each particle
      Calculate particle velocity
      Update particle position
   End
While maximum iterations or minimum error criteria is not attained
```

Algorithm 2.3: PSO Algorithm

PSO has proved to be efficient in solving Unconstrained Global Optimization and engineering problems [63, 64, 65].

2.6 SUPPORT VECTOR MACHINE

Support Vector Machine is a linear machine learning method based on structural risk minimization of statistical learning theory (STL) [6]. SVM shows good generalization skill, without the need of any prior knowledge. Furthermore, it does not suffer from the local minimum and it can handle noisy datasets.

The main objective of SVM is to find the optimum hyperplane to separate the two classes as shown in Figure 2.4 [66, 67]. We plot each data item as a point in n-dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. For this, it only needs a small amount of support vector quantities in order to define this hyperplane.

Given a training dataset (x_1, y1, x2, y2, x_N, y_N), where xi ∈ R^n is the n dimensional characteristic vector, y_i ∈ {−1,+1} is the class label and N denotes the total number of records from the training dataset, the hyperplane is defined by (w,b), where w is the weight vector and b is the bias. A new object x can be classified with the function g(x) = sign (w^T ·x + b) = sign (f(x)). Linear SVM is solved by formulating the quadratic optimization problem as:

$\min_{w,b}(w^T w /2)$, s. t. $y_i(w^T xi + b) \geq 1$ (i = 1,...,N)

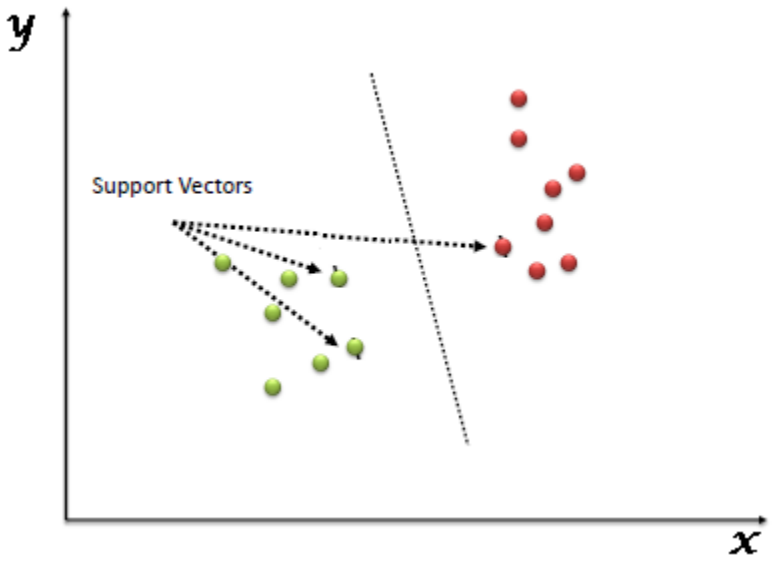

Figure 2.4: SVM Binary Classification

The dataset is not always linearly separable [6]. In these cases, we can introduce a slack variable $\xi_i \geq 0$ for each xi (i = 1,...,N). Furthermore, we can map the dataset into a higher-dimension feature space and try to find the hyperplane that linearly separates the mapped vectors. In other words, xi will be replaced with K(xi) where K provides the higher dimensional mapping (K is also called the kernel function). Usually, there are three main types of kernel functions: polynomial kernel function, radial-basis kernel function (RBF) and sigmoid function.

2.7 SUMMERY

In this Chapter we have reviewed literatures on web technology and security related issues to web technology. Moreover, the major types of security issues in web technology, characteristics of each type of treats and the methods how these treats have been tackled till now are presented. The various types of Nature Inspired Algorithms along with their objective are also discussed. Furthermore, Concepts which could facilitate the formulation of the solution domain are acquired.

Literatures that are revised in the initial sections of this research work happens to have a greater impact for the effort to know about web treats specifically phishing. Similarly, literatures written about Nature Inspired Algorithms in general and Support Vector Machine improved our level of understanding on domain specific topics

Chapter 3 : Related Work

As we discussed so far phishing is a criminal activity that has been a continual threat for different institutions and individuals who are using the Internet for different communication and financial transaction purposes. This is because of losses of different types of sensitive personal information mostly related to financial account details that leads individuals to loose their trust in different online based systems. This is due to the dynamic nature of phishing attack.

In this Chapter we will start by categorizing contributions related to our current research work under two classes namely nature inspired based and non-nature inspired based. The first section describes works based on algorithms which are inspired by the natural problem-solving behavior of social animals. On the other hand, the later describes various works derived by various mathematical problem-solving techniques and/or models.

Finally, the gaps of the reviewed works as well as the way the newly proposed solution fills those gaps is described briefly.

3.1 Nature Inspired Based Phishing Detection

The work in [68] shows how to implement the Ant colony optimization algorithm to detect e-banking phishing websites that will improve the correctly classified phishing websites. Besides that, it tries to describe how to enhance Particle Swarm Optimization (PSO) which finds a solution to an optimization problem in a search space, or model and predict social behavior in the presence of phishing websites that will improve the correctly classified phishing websites. Two publicly available datasets were used for testing the implementation: the "phishtank" from phishtank.com which is considered one of the primary phishing report collators and the Anti Phishing Working Group (APWG) archive. Java programing language was used to extract features, and store these in a database for quick reference. This was used in order to gather information about the strategies that are used by attackers and to formulate hypotheses about classifying and categorizing of all different e-banking phishing attack techniques. As a result of optimization on ACO and PSO the results were 89% and 91% of accuracy in correctly classifying a website as phishy or legal respectively.

In [69] a model has been proposed for predicting phishing attacks based on Artificial Neural Network (ANN). A Feed Forward Neural Network trained by Back Propagation algorithm is developed to classify websites as phishing or legitimate. The suggested model shows high

acceptance ability for noisy data, fault tolerance and high prediction accuracy with respect to false positive and false negative rates.

The dataset that have been used in this research consists of 1828 websites to extract 18 features using their own tool. The dataset is composed of 859 legitimate websites collected from Yahoo directory (http://dir.yahoo.com/) and starting point directory (http://www.stpt.com/directory/), and 969 phishing websites collected from Phishtank (http://www.phishtank.com/) and Millersmiles archives (http://www.millersmiles.co.uk/). The collected dataset holds categorical values, i.e., "Legitimate", "Suspicious" and "Phishy". These values are transformed to numerical values 1, 0 and -1 respectively. The dataset was split as 15% for validation, 15% for testing and 70% for training. The training dataset is used to train the network and to adjust the weights of the network, while the testing dataset remains unseen and it is used to assess the predictive performance of the model. After training, they run the network on the testing dataset. The error value on the testing dataset offers an unbiased approximation of the generalization error. The results showed that the best predictive performance was achieved when the number of neurons in the hidden layer was set to "2" and the learning rate was set to 0.7.

One of the problems in this research is that the criteria to identify the number of hidden layers is not mentioned clearly. Besides that determining the number of hidden layers was also difficult.

In [70] the authors propose a solution for phishing detection using BFOA. The aim is to identify phishing attack in three phases. In the first phase they have tried to collect 30 features that are going to be used for identification and classification of websites as legitimate and fake, the second phase was classification of websites using MCAR algorithm and in the last phase they implemented the BFOA optimization. The error rate of Bacteria Foraging algorithm results in less error rate when compared to the association classification using MCAR algorithm supervised with different optimization techniques like ACO and PSO with good number of iterations during optimization. It also predicts phishing websites with a better accuracy. The result shows that it has 94-95% prediction accuracy and 7-8% error rate. The research has made page style criteria which deals with the look and feel structure of the web along with social human factor for identifying phishing websites insignificant for the identification.

Most of the works on nature inspired based have common gaps which are:

- The features used for classification are common and manually selected which makes it difficult to have good generalization scheme on new sophisticated features,
- Performance optimization issues on the time that the classification will take have not been considered very well, and
- Using Nature Inspired along with machine learning method [66] has promising result compared with the results gained using only Nature Inspired Technique in terms of performance optimization and error rate.

3.2 Nature Independent Phishing Detection

3.2.1 Browser Based Methods

In this technique, the browser shows a warning to the user when it suspects an attack. For example, one of the techniques that is used by phishers is redirecting. For example, the URL http://www.google.com@warez.com may seem to be some section of the Google website. The browser also checks the security certificate of the website and reports if it finds anything suspicious.

In [71] the authors proposed a browser extension called AntiPhish which is used to protect inexperienced users against spoofed Website-based phishing attacks. AntiPhish is a plug-in tool which keeps track of users' sensitive information and prevents this information from being passed to a Website that is considered untrustworthy or unsafe. In order to identify whether a Website is a phishing site a text classification algorithm is used which works based on addresses used in a form. The algorithm compares a legitimate URL and IP address with the URL the page actually locates. AntiPhish focuses more on tracking sensitive information provided by a user identified a website as a suspect phishing site when the visual similarity value is above a pre-defined threshold.

One of the problems observed in this work is that it needs user support to capture and store sensitive information. For some users it might be better to provide a way where sensitive information is automatically captured and stored. It also suffers from the problem that legitimate reuse of credentials is also flagged as suspicious.

In [72] the authors presented an approach to detect phishing e-mails using link-based features. Phishing hyperlinks were categorized into the following general categories: 1) the actual link and the visual link in the e-mail are different, i.e., the hyperlink in the e-mail does not point to the same location as the apparent hyperlink displayed to the users; 2) The domain name in the hyperlink is substituted by the quad-tuple IP address; 3) Domain names used are manipulated to look similar to

the genuine DNS name the phishers are trying to forge; 4) The hyperlink is encoded so that it becomes very difficult to read, for example, unusually long hyperlink; and 5) when visiting the phishing hyperlink, it usually asks the user for various personal details like username, password, account number, SSN, etc. The link related features are divided to visible links, invisible links, and unmatched URLs. The proposed algorithm when used in conjunction with the proposed prototype of web browser will help the user to get notified of possible phishing attacks and will prevent them from opening suspicious websites.

This approach lacks visual similarity where phishers responded by compiling phishing pages with non-HTML components, such as images, Flash objects, and Java applets. For example, a phisher may design a fake page which is composed entirely of images, even if the original page only contains text information.

Both works reviewed in this section are vulnerable to poor set of results because of the complexity and nature of phishing attack. They are not an effective method of detecting phishing attack because most users do not pay attention to warnings and messages and press the OK and Accept buttons automatically.

3.2.2 Mathematical and Data Mining Based Methods

There have been different data mining techniques and also mathematical methods that have been suggested to tackle the problem of phishing attack.

The method proposed in [73] suggested using "CANTINA" which is content-based technique to detect phishing websites using the Term-Frequency-Inverse-Document-Frequency (TF-IDF) information retrieval measures. TF-IDF often produces weights that assess the word importance to a document by counting its frequency. CANTINA works as follows:

1. Calculate the TF-IDF for a given webpage.
2. Take the five highest TF-IDF terms and add them to the URL to find the lexical signature.
3. The lexical signature is fed into a search engine.

If the top N searching results have the current webpage, it is considered a legitimate webpage. If not, it is a phishy webpage. N was set to 30 in the experiments. If the search engine returns zero result, the website is labelled as phishy.

Hundred phishing URLs from PhishTank.com from November 17 to November 18, 2006 were chosen. All phishing URLs were selected within 6 hours of being reported. They also chose 100

legitimate URLs from a list of 500 used in 3Sharp's study of anti-phishing toolbars. All 200 URLs (100 phishing and 100 legitimate) were English language sites. The result shows that in comparing Basic-TF-IDF (Calculate the lexical signature based on the top 5 terms, submit that to Google, and check if the domain name of the page in question matches any of the top 30 results) to Basic-TF-IDF+domain, which is achieved by adding the domain name to the content of the webpage can significantly reduce the false positive rate (from 30% to 10%), but this comes at the cost of reduced accuracy (from 94% to 67%). This loss of accuracy is due to meaningless domain names that cause Google to return no results for some phishing sites. Comparing Basic-TF-IDF+domain to Basic-TFIDF+domain+ZMP, shows "zero results mean phishing" increases in accuracy (from 67% to 97%) without impacting the false positive rate. Thus, the Final-TF-IDF (Basic-TF-IDF+domain+ZMP) turned out to have the best results.

The limitation of this work is that some legitimate websites consist of images and so extracting the TF-IDF terms may not be accurate in this case. Moreover, this method is delayed in querying through a search engine and thus the user may have started in the disclosure of his/her personal information. Besides this approach does not deal with hidden texts which might be effective in detecting the type of the webpage.

In [74] fuzzy logic-based data mining technique has been deployed in order to detect e-banking phishing attack. The model works on multilayered approach, i.e., each layer should have its own rules. Moreover, the authors classify the website as very-legitimate, legitimate, suspicious, phishy or very-phishy. Two publicly available datasets were used to test the implementation: the phishtank from the phishtank.com and The Anti Phishing Working Group (APWG) Archive. The results indicate that the worst e-banking phishing website rate (all three layers have 10 input value) equals 83.7% representing very phishy website and the best e-banking phishing website rate (all three layers have 0 input value) is 16.4% representing very legitimate website rather than a full range, i.e., 0 to 100, because of the fuzzification process.

This approach works on multilayered approach where each layer should have its own rules. However, it was not clear if the rules were established based on human experience or extracted using an automated tool. Even though the authors classify the website as very-legitimate, legitimate, suspicious, phishy or very-phishy, they did not clarify what is the fine line that separate one class from one another

The work in [75] suggested a new way to detect phishing websites by capturing abnormal behaviors demonstrated by the websites. A structured website consists of "W3C DOM" features. The authors selected six structural features: abnormal URL, abnormal DNS record, abnormal anchors, server form handler, abnormal cookies and abnormal certificate in SSL. Support Vector-Machine classifier is used to determine whether the website is phishy or not. The classification accuracy in this method was 84%, which is relatively considered low.

The result produced in this work is relatively considered low and ignores important features that can play a key role in determining the legitimacy of the website, which explains the low detection rate.

3.3 SUMMARY

There have been several attempts to deal with phishing as presented in this chapter but there have been certain limitations that are uncovered by these researches. Researches are divided into two classes one that is based on nature inspired and the other are non-nature inspired based. Most of the researches that are not based on nature inspired will face difficulty in managing the dynamic nature of phishing attack. Especially browser-based methods are venerable to the various properties of phishing that are difficult to tackle.

Most nature inspired based works have better result compared to non-nature inspired but most of the works did not give attention on the performance of each algorithm in terms of the execution time and also the features that are used for identification are most commonly known.

Because of this there is still a huge suffering by most online based technologies specially that deals with different finical transactional and personal information. This study is intended to fill the gap observed on previous works and produce useful technique by combining the best features of nature inspired algorithms with support vector machine.

Chapter 4 : The Proposed System Architecture

4.1 OVERVIEW

The primary goal of this work is to design a model that enables the detection and reporting of web phishing attacks. The attack will be captured either on working Website or offline. After capturing the uniform resource locator (URL) of the webpage, it will be reported for the user and the user will have the option to block the web URL. In order to capture this illegal websites, we will use the Support Vector Machine Algorithm optimized with Nature Inspired Algorithms. This collaboration of the two algorithms will enable the detection much better perform due to dynamic nature of the phishing attack.

In order to achieve this goal, the model needs to take some features from the webpage visited by the user either from a direct access or an email attachment prior to classification of the website as phishy or legal. Once this necessary information about the website is taken and the classification is performed the user will get the phisher report.

One of the issues that is considered in this research is answering the question "how do this anti-phishing warnings actually be configured by users?". To make the research have good impact we have designed an option for the user to configure the warning levels in to three categories. This level express user's choice of active and passive phishing warnings. This category of warning is based on the study made to model warning [76] which provides how individuals may need to react on any issue of warning signs. The factors that affect this warning design are the following.

Attention and Comprehension: One of the major issues in warning management is how users notice the indicators and have the perception of being in some form of attack. This means the understanding of users of what the indicators mean and know what they are supposed to do when they see the indicators.

Attitudes and Motivation: The other issue is how users believe in the warning indicators they receive and how they are motivated to take the recommended actions. We need to consider how users take seriously perform those actions.

We have considered the issue of warning message about the phisher report to be configured in three main level (high, medium and low) which depends on user's intellectual level.

4.2 FUNCTIONAL MODEL/ARCHITECTURE

4.2.1 Overview

This Section presents a general architecture of the proposed model which is designed to achieve the primary design goal as stated in the previous section. The components in this model could be implemented at any level of communication standard: only on the server side or adding on a client machine for some of the components. The functional architecture is shown in Figure 4.1 which is further explained in the subsequent sections.

There are few assumptions that are made in designing this model: The user is expected to apply the client-side components in his/her machine; the user is also expected to configure the warning mechanism for the reporting of the website.

One of the issues that is covered in this architecture is the issue of generating a new model on the arrival of new data sets. This means the model will be built repeatedly after the initial training is accomplished. This helps the system to make sure that it grows with the knowledge of new form of attack from the relearning process. This is advantageous when dealing with very large or non-stationary data. In the case of non-stationary data, batch algorithms will generally fail if ambiguous information, e.g., different distributions varying over time, is present and is erroneously integrated by the batch algorithm. In order to achieve this feature of our learning model we will track the dataset that is stored in URL list to continuously train itself.

The other issue that this system provides an API that helps the reporting process coordinating with phishtank website. This means that we will design a platform that enables the phishing reporter component submit the URL of the phishy web address with some extra information about the page. This helps for different study on the phishing page and achieve better accuracy in the retraining process since phishtank performs evaluation of submitted URL before they officially accept and post the URL as phishing page.

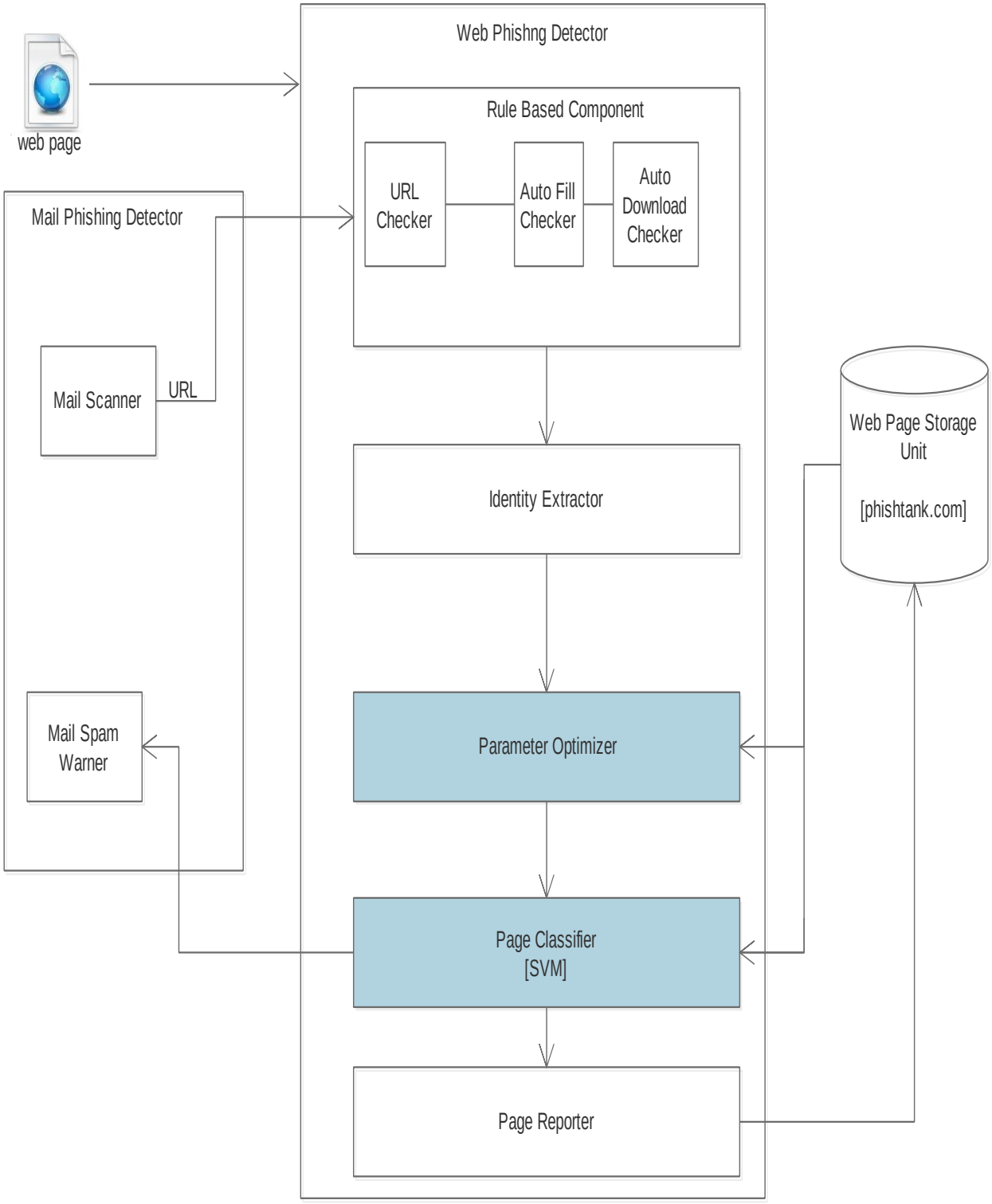

Figure 4.1: Functional Architecture

4.2.2 Components

a. Mail Phishing Detector

One of the major recent findings is that there has been a dramatic increase in phishing in which victims are tricked by spoofed emails. This phishing attack starts by sending an e-mail that seems authentic to potential victims urging them to update or validate their information by following a URL link within the e-mail.

In order to tackle this problem, we have proposed a model that comprises a component that is able to scan inbox and spam messages from user's mail once the user configures his/her preference. Once the user has provided his/her preference, the system is able to login and scanning user's mail on the search for the existence of URL on the subject, and body of the inbox. Assumption that is made about this component is to work as a background process on user's computer once the user configures his/her preference and protection level.

1. **Mail Configuration:** In order to perform the mail phishing process, the user is expected to initialize by providing personal information regarding his/her email address and password. Besides the user's email information, the users are expected to provide the security measure level.

   ***Level 1*:** Users will choose this level if they want the system do nothing on the inbox but highlight the email and put marker that shows detail information about the analysis result.

   ***Level 2*:** Besides highlighting the email and putting marker that shows detail information about the analysis result, move to spam and also optionally disable the link to be opened.

   ***Level 3*:** Delete the mail.

2. Mail Scanner: The mail scanner will work as a background process and scans every time a new email arrives at the user's inbox or spam. The mail scanner is described in Algorithm 4.1.

Input: Email

Output: Find the existence of URL

```
Function ScannMail ()
    Do
      # extract user mail information from mail
        Get the mail as a User Mail
        Extract the username
```

```
        Extract the password
    If (Account extracted)
        Try the login
        # extract information from email inbox /spam
        Get the inbox of user
        Scan for URL from the subject or body of the mail
      If (URL is not null)
         Return URL
      Else
        Do nothing
```

Algorithm 4.1: Mail Scanner Algorithm

3. **Mail Classifier:** Once the mail scanner has reported the existence of URL on new arrived inbox a detail analysis will be performed on the page. This component is similar with the page classifier described in section 4.2.2.4 except we will make it to work as a background process.

4. **Mail Spam Warner**: Once the URL scanned is classified by the mail classifier this component is responsible to warn the user about the page information that is found in user inbox. This warning type depends on the user's preference setting**.**

b. Rule based Component

This component is designed to increases the efficiency of the detection because the detection process has to be made with minimum amount of time.

1. **URL Checker**: This component tries to verify URL based on the lexical features. Features that describe lexical patterns of malicious URL are extracted to further analysis the nature of the page. The first component of the URL Checker is the Feature Extractor which has the responsibility of extracting Lexical information from the URL like the length of the URL, finding @ symbol on the URL, getting number of sensitive words, Top level Domain name(TLD) placing, port number, double bar and checking for the existence of IP on the URL. This feature is listed and described as follows:

a) Length of URL: Phishers can use long URL to hide the doubtful part in the address bar. For example:

http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&

dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html.

Most studies [77, 78, 79] showed that if the length of the URL is greater than or equal to 54 characters then the URL is classified as phishing.

b) Number of dots in URL: This feature counts the number of dots in the URL. Phishing pages tend to use more dots in their URL than legitimate sites.

c) Suspicious special symbol on URL: This feature checks if a page's URL contains an "at" (@) or the domain name has a dash (-). The @ symbol in a URL causes the string to the left to be disregarded, with the string on the right treated as the actual URL for retrieving the page. Combined with the limited size of the browser address bar, this makes it possible to write URLs that appear legitimate within the address bar, but actually cause the browser to retrieve a different page. Dashes are also not often used by legitimate sites.

d) Number of sensitive words in URL. In [79], Garera *et al.* summarized a set of eight sensitive words that frequently appear in phishing URLs, and we create this feature counting the number of the eight sensitive words that are found in a page URL which are listed as follows: "confirm","account","banking","secure","ebayisapi","webscr","login","signin". This is a numeric feature with a range of 0 to 8.

e) Out-of-position top level domain (TLD): This feature checks if a TLD appears in an unusual position in the URL. An example is http://cgi.ebay.com.ebaymotors.732issapidll.private99dll.qqmotorsqq.ebmdata.com, in which we see the TLD "com" in a position usually not for TLDs in the hostname.

f) Using Non-Standard Port: This feature is useful in validating if a particular service (e.g., HTTP) is up or down on a specific server. Several firewalls, Proxy and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only open the ones selected. If all ports are open, phishers can run almost any service they want and as a result, user information is threatened. The most important ports and their preferred status are shown in Table 4.1. If the port status is from a closed status then the page is categorized as phishy page.

Table 4.1: Ports need to be Checked

PORT	Service	Meaning	Preferred Status
21	FTP	Transfer files from one host to another	Close
22	SSH	Secure File Transfer Protocol	Close
23	Telnet	provide a bidirectional interactive text-oriented communication	Close
80	HTTP	Hyper test transfer protocol	Open
443	HTTPS	Hypertext transfer protocol secured	Open
445	SMB	Providing shared access to files, printers, serial ports	Close
3306	MySQL	Access MySQL database from web.	Close
3389	Remote Desktop	allow remote access and remote collaboration	Close

g) ***Redirecting using "//":*** The existence of "//" within the URL path means that the user will be redirected on another website. An example of such URL is: http://www.legitimate.com// http://www.phishing.com. We will examine the location where the "//" appears. We find that if the URL starts with "HTTP", that means the "//" should appear in the sixth position. However, if the URL employs "HTTPS" then the "//" should appear in seventh position.

h) Using the IP Address: If an IP address is used as an alternative of the domain name in the URL, such as "http://10.5.44.45/phsihng.html", users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link "http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html". By using the features that are extracted from the URL first the system will perform the classification (phishing or legitimate) then the phishing URL will be reported to the user and submitted to a global repository.

Once the URL information is extracted from the URL object the value of each attribute is checked against abnormality to decide whether the URL represents a legitimate or phishing webpage. Finally, once the classification has been performed the user will be warned about the security issue regarding the webpage based on the configuration level the user sets. If the URL is found to be phishing it will be necessary to remove from the history of the browser so that the user will not suffer from autofill of web address on searching. The working model of this component is shown on the Algorithm 4.2.

Input: A Page URL

Output: Phishing or legitimate page message, browser history removal function call

```
Function main ()
    Do
        PerformAction ()
Function URLInfoExtractor()
   Do
    # extract each page information from URL
       Get the URL from the address bar
       Define a list URLinfo
       Declare variables URLlength, URLdots, URLsymbol, URLsenstivewords, URLtdlplace,
         URLportno, URLdoublebar, URLIPadress
         URLlength ← Extract the URL Length
         URLdots ← Extract the Number of Dots
         URLsymbol ← Extract if an @ symbol exist
         URLsenstivewords ← Extract the Number of Sensitive Words
         URLtdlplace ← Extract The Top domain place
         URLportno ← Extract the Port Number
         URLdoublebar ← The number of double bar
         URLIPadress ← The existence of ip in URL
         Add each value to the URL info list
     Return URLInfo
Function URLBasedPageClassifier
  # by using the extracted info about page perform the classification
  Do
    pageType ←"legitimate "
      # URLInfoExctracto method will return array of URL info
     Declare a pageInfo list
    Array pageInfo ← URLInfoExtractor()
    If (pageInfo [first element] > =54)
        pageType ←phishing
```

```
    If (pageInfo [second element] ==Many Dot)
        pageType ← phishing
    If (pageInfo [third element] > =have @ symbol or dashed symbol (-))
        pageType ← phishing
    If (pageInfo [fourth element] > =8)
        pageType ←phishing
    If (pageInfo [fifth element] ←changed TDL place)
        pageType ←phishing
    If (pageInfo [sixth element] == unusual port number)
        pageType ← phishing
     If (pageInfo [seventh element] == Double Bar exists)
        pageType ← phishing
    If (pageInfo [eighth element] == IP exists)
        pageType ← phishing
     Return the pageType
Function PerformAction
     Do
    message ← URLBasedPageClassifier ()
    If (message == "phishing")
      Warn user based on their configuration
      Call removeBrowserHisory (the URL)
    Else
      Continue to the SVM classifier component
```

Algorithm 4.2: URL Checker Algorithm

2. Auto Fill Checker: Autofill [80] is a browser feature that lets one to store user credentials like name, e-mail, phone number, etc. so that one doesn't have to fill it manually again and again. Although most users see autofill feature as a convenience that provides ease while filling out credentials, it could also lead to big security risks leading to financial losses and a threat to personal data. The developer of a site can add hidden fields to a page, which is actually not "really hidden" from users but drawn outside the visible screen. Due to Autofill feature, user's browser will automatically fill out fields which you can see and also the ones which cannot. Our design tries to

address this issue by considering the number of fields that the autofill try to submit which is shown in Algorithm 4.3:

```
Input: State of Autofill
Output:  Send Null value
Function ScannMail ()
    Do
      # Check autofill status and if on the run this functionality
       Declare autofillstatus
        autofillstatus ← "on/off"
       If(autofillstatus is in off state)
            Do Nothing
       Else
        autoFillEnteries ←Number of entries from auto fill
        visiblePageEneries ←Number of visible entries
       If (autoFillEnteries  ! =  visiblePageEneries)
            Send the null values of entries on the autofill
```

Algorithm 4.3: AutoFill Protection Algorithm

Once the autofill entries are counted and the number of visible entries on the page are inspected the autofill protection unit will try to send null values of non-entries on the page which might be invisible form elements designed to collect user information.

3. Auto Download Checker: A drive-by download [81] is a malicious phishing website that attempts to install malware onto one device without asking for permission first. Most of the time these types of difficulties arise when we try to access webs from our Android based smart phones which tries to install apk files without our permission which leads to leak of information. In our design we have tried to address this issue by first checking the operating system, basically Android or Windows, then inspect the page if it is trying to download any file without permission. This is shown in Figure 4.4.

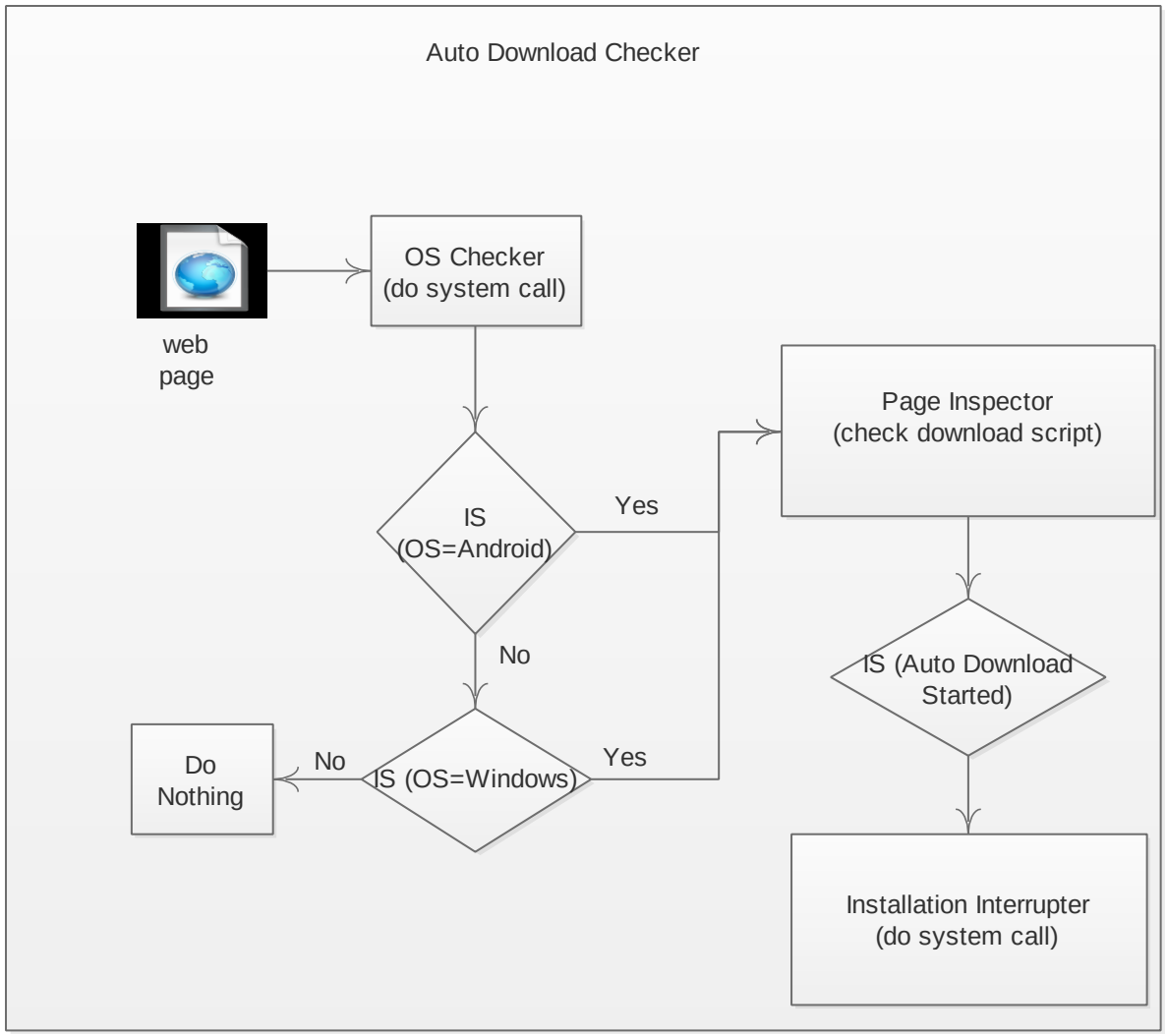

Figure 4.2: Auto Download Checker

b. Identity Extractor

There are many characteristics and indicators that can distinguish a legitimate website from the phishing one. This system managed to gather 28 phishing features [68, 69, 70] and indicators and we have clustered them into six. The identity extractor we have proposed will try to extract these 28 features of a specific webpage. The Algorithm is shown in Algorithm 4.4.

Input: URL

Output: 28 features

#This variable holds the value of abnormality yes or no

Declare variable *abnormal*

Function main ()

```
    Do
      # Check autofill status and if on the run this functionality
       Declare featureVector list to store the 28 features
      featureVector  ← allFeatures()
      Convert each value in featureVector to -1 and 1 for yes and no respectively
      Return featureVector
Function allFeatures ()
 Do
     For each function
      Add featureVector ← result of call to each function
     End
# URL & Domain Identity Features
Function usingIp ()
  Define a regular expression for number checking on string
  Compare against the URL
  If (Number Exist on URL)
    abnormal ← yes
   Return abnormal
Function abnormalRequests ()
  While end of file
    Extract each objects form the page [image, video,favcon, and link]
    Check each objects domain loaded from
    If (domain different form the URL domain)
       abnormal ← yes
   Return abnormal
Function abnormalDNS ()
  While end of WHOIS record
    Check the hostname exist
    Check the identity match
    Check for valid hostname
    If (hostname doesn't have valid identity or hostname)
```

```
        abnormal ← yes
    If (hostname doesn't exist)
        abnormal ← yes
    If (hostname age less than 6 -12 month)
        abnormal ← yes
  Return abnormal
Function outOfPostionBrandName ()
  Define a regular expression for brad name
  Compare against the URL
  If (Mismatch brand name position on URL)
    abnormal ← yes
  Return abnormal
Function subDomain ()
  Define a regular expression URL domain name
  Compare against the URL subdomain
  Count the number of subdomain
  If (Mismatch subdomain on URL)
    abnormal ← yes
  Return abnormal
Function subDomain ()
  Define a regular expression for search q domain
  Compare against the URL domain
  If (Mismatch domain on URL)
    abnormal ← yes
  Return abnormal
#Security & Encryption Features
Function usingSSLCertificate ()
  Request for the certificate of the page
  Extract identity of the page
  Extract certificate authority of page
  Check if identity matches
```

```
  If (Mismatch identity mismatch)
    abnormal ← yes
  If (certificate authority domain == attacker domain)
    abnormal ← yes
  Return abnormal
Function CookiDomain ()
  Request for cookie of the page
  Cheek the domain of the cookie
  Check if domain matches
  If (Mismatch in domain)
    abnormal ← yes
#Source Code and JavaScript Features
Function redirectPage ()
   Count the number of redirect pages by using the position of //
   If (number > 2 and < 4)
     abnormal ← yes
   Return abnormal
Function mouseHover ()
   Find the link objects in the page
   Check the mouse hover attribute
   If (attribute == null)
     abnormal ← yes
   Return abnormal
Function serverFormHandler ()
   Extract the form tag
   Check the action attribute
   If (attribute == null or attribute == blank)
     abnormal ← yes
   Return abnormal
#Page Style and Content Feature
Function UsingFFP ()
```

```
  Extract the Specific tag on every page [iframe, form, popup]
  If (exist one page a popup or iframe)
    abnormal ← yes
  If (exist a form with hidden fields)
    abnormal ← yes
Function diableRightClick ()
  Load the JavaScript of the page
  Check the right Click Property
  If (property == disabled)
    abnormal ← yes
  Return abnormal
Function gremmerChecker ()
  Load the grammar Checker
  Load the main page of the URL
  Check for grammar and Spelling Check
  If (result == 25% error)
    abnormal ← yes
  Return abnormal
#Web address bar features
Function checkAddtionalCharachter ()
  Define a regular expression for checking characters [-, @, hexadecimal char code]
  Compare against the URL
  If (Mismatch on URL)
    abnormal ← yes
  Return abnormal
Function urlLength ()
  Extract the length of URL
  If (length >= and <= 75)
    abnormal ← yes
  Return abnormal
Function urlSimlarchar ()
```

```
  While end of URL
    Check existence of [0, 1, 6,8]
  If (exist)
    abnormal ← yes
  Return abnormal
#Search engine based features
Function pageRank()
  Request google for page rank
  If (pagerank < 0.2)
    abnormal ← yes
Function googleIndex()
  Request google index of the page
  If (not indexed)
    abnormal ← yes
```

Algorithm 4.4: Identity Extraction Algorithm

1. URL & Domain Identity Features

a) ***Using IP address*:** Using IP address in the hostname part of the URL address means user can almost be sure someone is trying to steal his/her personal information. Not only using IP address the IP may not be resolvable to any hostname.

b) ***Abnormal request URL***: A webpage usually consists of a text and some objects such as images and videos. Typically, these objects are loaded to the webpage from the same domain where the webpage exists. If the objects are loaded from a domain different from the domain typed in the URL address then the webpage is potentially suspicious.

c) ***Abnormal URL of anchor***: Similar to "abnormal request URL" but for this feature the links within the webpage might refer to a domain different from the domain typed on the URL address bar. As we have seen on abnormal request URL this feature is treated exactly as "Request URL".

d) ***Abnormal DNS record***: A full DNS record usually has identity relevant information. For phishing sites, either the record of the hostname is not found in the WHOIS database or the claimed identity is not contained in the record.

e) ***Domain Registration***: Based on the fact that a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance. In our dataset, we find that the longest fraudulent domains have been used for one year only.

f) ***Age of the Domain*****:** One main technique to identify phishing pages is by looking the age of the domain name. Many phishing sites have domains that are registered only a few days before phishing webpage are accessed by users or emails that contain this URLs are sent out. WHOIS search will be used to implement this heuristic. This heuristic measures the number of months from when the domain name was first registered. If the page has been registered longer than 6-12 months, it will return +1, deeming it as legitimate, and otherwise returns -1, deeming it as phishing. If the WHOIS server cannot find the domain, the heuristic will simply return -1, deeming it as a phishing page. The other issue related to the domain age is weather these pages have high traffic since they are visited regularly. Phishing websites often have a short life thus their web traffic either does not exist or is ranked below the limit that gives it the non-legitimate status.

g) ***Sub-domain(s) in URL***: Another technique used by phishers to deceive users is by adding sub-domain(s) to the URL thus users may believe that they are dealing with a credited website.

h) ***Out-of-position brand name***: The vast majority of companies put their brand name into their domain name. Typically, the brand name appears in the domain string as the second-level or third-level domain. Phishing sites, however, are always hosted on compromised or newly registered domains. To make these sites look trustworthy, attackers sometimes include brand names or domain names of the victim sites in their phishing URLs, causing an out-of-position brand name.

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

j) **Links in <Meta>, <Script> and <Link> tags:** Given that our investigation covers all angles likely to be used in the webpage source code, we find that it is common for legitimate websites to use <Meta> tags to offer metadata about the HTML document; <Script> tags to create a client

side script; and <Link> tags to retrieve other web resources. It is expected that these tags are linked to the same domain of the webpage.

2. Security & Encryption Features

a) ***Using SSL Certificate***: For SSL-enabled phishing sites, public key certificates are employed. In many phishing attacks, the Distinguished Names (DN) in their certificates are inconsistent with the claimed identities.

b) ***Certificate Authority***: Usually SSL certificate authorities (CAs) will not issue a certificate to fraudulent phishing. Such websites usually do not use SSL encryption. Most of the time Phishing website could try to use the certificates issued by certification authorities in the domain owned by the attacker.

c) ***Abnormal Cookie***: Cookies [82] are deposited by a web server to its client, so that the client information is sent to the server when the browser visits it again. Therefore, a cookie is usually bound to its web server's domain. For a phishing site, its cookies either point to its own domain, which is inconsistent with the claimed identity, or point to the real site which is inconsistent with its own domain.

3. Source Code & Java script Features

a) ***Redirect pages***: This feature is commonly used by phishers by hiding the real link they ask users to submit their information to a suspicious website.

b) ***On Mouse over to hide the Link:*** Users may get a phishing email that contains an HTML page attachment. When we open the page in a browser, the page will be loaded using a local client-side URL. If the page contains a form with submit button, hovering over the "Submit" button shows an authentic PayPal URL, which makes no sense. These are JavaScript files loaded by this phishing email, which contained code that hijacked user clicks. This JS code was set to replace any requests to paypal.com with the malicious phishing URL, right after the user clicked the link. Hovering the URL would not do anything, and the browser shows the correct PayPal link.

c) ***Server Form Handler (SFH):*** Once the user submits his/her information, that information will be transferred to a server to be processed. Normally, the information is processed from the same domain where the webpage is being loaded. Phishers resort to make the server form handler either empty or the submitted information are transferred to different domains.

4. Page Style & Contents Features

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

b) ***Grammatical and Spelling Errors***: Most of the time individuals who develop phishing websites are not worried about the entire content of the page. Since these developers require the page for short period of time, they make different kinds of grammatical and spelling errors when they produce the visual imitation.

c) ***Using forms:*** Phishing attacks are usually accomplished through HTML forms. This feature checks if a page contains potentially harmful HTML forms. To satisfy our definition of harmful, a webpage is required to have all of the following: 1) an HTML form, 2) an <input> tag in the form, 3) keywords related to sensitive information like "password" and "credit card number" or no text at all but images only within the scope of the HTML form, 4) a non-HTTPS scheme in the URL in the action field or in the webpage URL when the action field is empty.

d) ***Using pop-ups Windows***: It is unusual to find a legitimate website that asks users to submit their credentials through a popup window. So most of the time a phishing page manipulates popup windows for retrieving private information from users.

e) ***Disabling right-click***: Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as "Using on Mouse Over to hide the Link". Nonetheless, for this feature, we will search for event in the webpage source code and check if the right click is disabled.

5. Web Address Bar Features

a) **Long URL Address**: by using a long URL phishers resort to hide the suspicious part of the URL, which may redirect the information submitted by the users or redirect the uploaded page to a suspicious domain.

b) **Replacing similar char for URL**: One of the techniques used to lure users visit a fake Website is by using similar characters so that users cannot identify weather they are accessing a legitimate webpage. For example www.lion.com can be tricked by replacing l with number one

and this will result in www.1ion.com or other example could be replacing 0 with O character such as www.liOn.com with www.li0n.com.

c) **Adding a prefix or suffix**: Phishers deceive users by reshaping the URL to look like legitimate one. A technique used to do so is by adding prefix or suffix to the legitimate URL so users might not notice any difference.

d) **Using the @ Symbols to confuse**: The "@" symbol leads the browser to ignore everything prior to it and redirects the user to the link typed after it.

e) Using hexadecimal char codes: This attack has an HTML attachment with a JavaScript snippet, and the content is encoded through hexadecimal escape characters, therefore no links are visible, but when opened, it presents a locally-generated phishing page with login instructions.

6. Search Engine based Features

a) ***PageRank***: PageRank is a value ranging from "0" to "1". PageRank aims to measure how important a webpage is on the Internet. The greater the PageRank value the more important the webpage.

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

c. Page Classifier

1.A Support Vector Machine

Our page classifier uses Support Vector Machine (SVM) [6], a well-known algorithm for classification. The main objective of SVM is to find the optimum hyperplane to separate the two classes. Since a webpage is either faked or authentic, phishing detection is by nature a binary classification problem. For this, it only needs a small amount of support vector quantities in order to define this hyperplane. Given a training dataset $(x_1; y_1)$; $(x_2; y_2)$; :::; $(x_N; y_N)$, where $x_i \in R^n$ is the n dimensional characteristic vector, $y_i \in \{-1; +1\}$ is the class label and N denotes the total number of records from the training dataset. SVM classifier has the ability of finding a hyperplane in the *n*-dimension space, which separates the dataset into two classes such that all vectors on one side of the hyperplane have label 1 while those on the other side have label -1. Given this property, SVM is

used to classify vectors without prior knowledge of their distribution. The training process enables SVM to develop a classification model, based on which a trained SVM is able to label a vector from a test set. The hyperplane is defined by (w; b), where w is the weight vector and b is the bias. A new object x, can be classified with the following function $g(x) = sign(w^T * x + b) = sign(f(x))$.

We make use of SVM as the page classifier. Its input is a 28- dimension vector representing a webpage's 28 structural features. It outputs a label 1 indicating a phishing page or a label -1 indicating an authentic one.

2.Feature Vector Generation

To facilitate SVM based classification, we quantify those features into vectors which are shown in Table 4.2. In our feature generation process the webpage file is represented as P and each of the 28 features are represented as F_i where i is $1 \geq i \leq 28$. Finally we get a page p having 28 feature vector: $P<F_1\ F_2,\ F_3,\ F_4 \ldots\ldots\ F_{28}>$

Table 4.2: Feature Vector

Features		Criteria	Phishy (-1)	Legitimate (1)	Suspicious (0)
URL & Domain Identity Features					
F1.	Using IP address	Unresolvable IP	√		
		Using IP	√		
		Otherwise		√	
F2.	Abnormal request URL	% of request URL of anchor < 22%		√	
		% of URL of anchor ≥ 22% and ≤61%			√
		Otherwise	√		
F3	Abnormal URL of anchor	% of URL of anchor < 31%		√	
		% of URL of anchor ≥ 31% and ≤67%			√
		Otherwise	√		
F4.	Domain Registration	Domain expires <= 1 year	√		
		Otherwise		√	
F5.	Age of the Domain	Age ≥6month		√	
		Otherwise	√		
F6.	Sub-domain(s) in	Dotes in the domain = 1		√	
		Dotes in the domain = 2			√

	URL	Otherwise	√		
F7.	Out-of-position brand name	Brand name position is in level 2 || 3		√	
		Otherwise	√		
F8.	Abnormal DNS record	No DNS record	√		
		Otherwise		√	
F9.	Favicon	If favicon loaded from different domain	√		
		Otherwise		√	
F10.	Links in <Meta>, <Script> and <Link> tags	% of links in this tags <17%		√	
		% of links in this tags ≥17% and ≤ 81 %			√
		Otherwise	√		
Security & Encryption Features					
F11.	Using SSL Certificate	claimed identities does not appear in the certificate	√		
		SSL no applied			√
		Otherwise		√	
F12.	Certificate authority	certificates issued by domain owned by the attacker	√		
		Certificate issued by known authorities		√	
		Otherwise			√
F13.	Abnormal cookie	if any different domain exists in cookies	√		
		If page have no cookie			√
		other		√	
Source Code & Java script Features					
F14.	Redirect pages	No of redirect pages <= 1		√	
		In between 2 and 4			√
		Otherwise	√		
F15.	On Mouse over to hide the Link	Mouse over with link			√
		Otherwise		√	
F16.	Server Form Handler (SFH)	SFH “about: blank” or Is Empty	√		
		SFH use different domain			√
		Otherwise		√	

Page Style & Contents Features					
F17.	Iframe	Use iframe	√		
		Otherwise		√	
F18.	Grammatical and Spelling Errors	% of page grammar/spelling =25%	√		
		Otherwise		√	
F19.	Using forms	If forms are used			√
		Otherwise		√	
F20.	Using pop-ups windows	Popups contain text field	√		
		Otherwise		√	
F21.	Disabling right-click	Right click disabled	√		
		Otherwise		√	
Web Address Bar Features					
F22.	Long URL Address	URL length < 54	√		
		URL length ≥ 54 and ≤ 75			√
		Otherwise		√	
F23.	Replacing similar char for URL	Existence of replicable characters			√
		Using numbers instead of character		√	
		Otherwise		√	
F24.	Adding a prefix or suffix	Domain name includes (–)symbol	√		
		Otherwise		√	
F25.	Using the @ Symbols	URL having @ symbol	√		
		Otherwise		√	
F26.	Using hexadecimal char codes	Using hexadecimal char code	√		
		Otherwise		√	
Search Engine based Features					
F27.	PageRank	PageRank < 0.2	√		
		Otherwise		√	
F28.	Google Index	Webpage indexed by Google		√	
		Otherwise	√		

***d*. Decision Optimizer**

Optimization [1] is nothing but selection of a best element from some set of available alternatives. An optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. In most of training the classier we face the training data is often not linearly separable, so SVM introduces the concept of "kernel induced feature space" which maps the data into a higher-dimensional space where the data is separable into two classes. Typically, such a mapping space would cause problems computationally. But, the results of SVM can be made more robust by choosing an appropriate generalizing grade which are represented by the regularization parameters "soft margin constant" (C) and "kernel parameter" (K). The goal is to find these parameters such that they minimize the error, that is, the misclassification penalty.

In this thesis we propose to apply Firefly Algorithm [83], a popular algorithm used for optimization problems. By Firefly algorithm, it was established that, the more number of comparisons between the fireflies to find the best location in the swarm, the better the results can be. This comparison is based on the light intensities of the fireflies, with which they glow. This concept makes use of the fact that, with more brightness of a firefly, the more is the attraction with the other fireflies, which reduces the distance between them. Thus, the lower the distance, the better the attraction and more is the scope of finding the best position in the pool of fireflies, which correspondingly gives the best parametric value, which can be fed as an input to the classifier.

In this proposed model we take in to consideration the advantage of the Firefly algorithm. By this method, the parameters of SVM are obtained by the firefly algorithm and the newly obtained parameters are fed as an input to the SVM classifier. The logic used over here is that each firefly is compared to every other firefly in the swarm, and based on the brightness, one best location is chosen. This best location corresponds to the parameters of SVM, which are now optimized. The strategy is that corresponding to the light intensity, which was realized to update the new positions of the fireflies, the pool is searched for the best value of the brightness, towards which all the fireflies move with some randomness then the values are updated. The algorithm is shown in Algorithm 4.5[84, 85]:

Input: Object function (f)
Output: Finding Optimized Values from search space

Begin:

Initialize

Alpha Beta Gamma Delta

Maximum number of generations (g)

Number of fireflies (n)

Range of values for SVM parameters

for (count of firefly ← 1 to g)

for (count of SVM) ← 1 to n)

Evaluate the performance of SVM classifier with parameters as obtained by the firefly chosen by (count of SVM).

Rank (sort) the fireflies in the increasing order of their light intensity.

Move all the fireflies to better locations as

$x_n(f) \leftarrow x_0(f)*(1-\beta) + x_0(f) * \beta + \alpha *(rand - 0.5)$

$y_n(f) \leftarrow y_0(f)*(1-\beta) + y_0(f) * \beta + \alpha *(rand - 0.5)$

Reduce the value of randomness parameter alpha by multiplying with parameter delta.

Use the optimized parameters with the highest intensity to train the SVM classifier

Algorithm 4.5: Optimization of SVM parameters with firefly Algorithm

e. Page Reporter

Once the classification task has finished the classifier will result in a numeric value of 1 and -1 then these values are changed to phishy and legitimate respectively. The main construction of this module is provided through user's configuration where the user configures preference information on how the warning report should be displayed. The other task of this page reporter is to submit the page found as a phishy to a storage unit for further study. We consider three level of configuration:

Level 1: The user will choose a strong warning that may go up to closing the page where the user is visiting currently if it is found to be phishy.

Level 2: The user will choose a medium warning that may go up to stopping the user from visiting the page by displaying popup message dialog when the current page is found to be phishy.

Level 3: The user will choose a low warning that only displays a warning message to users when s/he is visiting a phishy page.

4.3 SUMMARY

In this Chapter, we presented the architecture and algorithm of the proposed webpage analysis for the detection of phishing activities. In this architecture we decided to consider the detection in two mechanisms: The first is to detect web phishing that are performed by email and others outside mail when the user tries to open a specific page.

The architecture of the proposed system contains five key components: Rule Based Component, Identity Extractor, Page Classifier, Decision Optimizer, and Page Reporter. The model takes webpages information coming either from the mail scanner that is assumed to work as background process or from a direct access of a URL typed by the user on the browser address bar. Once the page URL is captured the URL based component tries to make a classification of the page based on the URL information extracted by URL checker. The page is also checked against browser autofill feature and auto download starting feature. If the URL based components are unable to detect the page as phishy it is passed to the page classifier that is based on machine learning technique.
The first component which is the identity extractor tries to collect various information regarding the page. This page information are classified into six major categories: URL & Domain Identity Features, Security & Encryption Features, Source Code & Java script Features, Page Style & Contents Features, Web Address Bar Features, and Search Engine based Features

Once this page information is extracted, feature vectors are generated for support vector machine in order to perform the classification. The classification process starts once the feature vectors are generated and the SVM is trained with this webpage having this features in the storage unit. In this proposed model we take into consideration the advantage of the Firefly algorithm. By this method, the parameters of SVM are obtained by the firefly algorithm and the newly obtained parameters are fed as an input to the SVM classifier.

Once the page classification is done the page reporter component will inform the user about the webpage s/he is accessing based on the user's configuration level.

Chapter 5 : Implementation and Evaluation

5.1 INTRODUCTION

In this Chapter, the developed prototype for phishing detection and reporting and its evaluation is presented. Subsequent sections present development tools, the working prototype model of different components, experiments and performance evaluations, evaluation parameters and discussion of the result.

Our main goal with this implementation is to prove the practicability of having such a framework with an appropriate user management module. It is also to establish a starting point for browser development with the integration of phishing detection and warning components through user configuration. This helps as a means of evaluation against future improvements that would normally be a result of enhancements made to the various components of the model and the browser.

There are few assumptions made during the development on the browser for testing and evaluating the proposed model.

5.2 ASSUMPTIONS ON IMPLEMENTATION AND EVALUATION

One assumption that is made during the development of the model is distributing the client service process where the phishing detection service will be consumed as a web service. Web services can be generally regarded as functions of applications exposed over the web using standardized message formats and typically interfaced to other software using traditional APIs, although "message-centric" usage of such services is also possible and may be favored by certain technologies. Representational State Transfer (REST) has proved to be a popular choice for implementing Web Services [86].

REST describes a set of architectural principles by which data can be transmitted over a standardized interface (such as HTTP). REST does not contain an additional messaging layer and focuses on design rules for creating stateless services. A client can access the resource using the unique URI and a representation of the resource is returned. With each new resource representation, the client is said to transfer state. While accessing RESTful resources with HTTP protocol, the URL of the resource serves as the resource identifier and GET, PUT, DELETE, POST and HEAD are the standard HTTP operations to be performed on that resource.

The other assumption that is made on the deployment of the model is the initiation of user trade when the user starts the client application that lead to the scanning of the URL and generation of analysis report.

5.3 DEVELOPMENT TOOLS

For the development of different components and prototypes we have used various tools and frameworks. The tools and languages/frameworks with their respective role is shown in Table 5.1.

Table 5.1: Language and Tools used on Prototyping

Tools	**Additional Framework**	**Used For**
NeatBean IDE with Scene Builder	Java Fx	The development of a sample browser Client-side application
>>	Java Warning Modules	Warning Management
Python IDL	Python language URLLib BuitfullSoap	Page Information Extraction Mail Scanner Auto Download Component
>>	sklearn	For SVM implementation For Fear Fly algorithm implementation
Sublime	JavaScript Google API Clinet Libraray Ajax	Mail Configuration and Labeling
>>	PHP Phish tank API	For submitting phishy URL for phishtank
Data Representation	JSON	Transferring data between components
Ruby	Rubyesque Gmailgem	For Gmail scanning Gmail Marking
MySQL 5.7	Database	For storage

5.4 EXPERIMENTAL SETTING

The experimentation has been done on a laptop and desktop PC with Windows 10 operating system, 2.40 GHz Intel CPU, 8 GB RAM and 1TB hard disk. MySQL database, Java NetBeans, Python 3.6.0, Ruby 3.0 and necessary packages were installed and configured for the development and testing of the developed model.

5.5 Data Collection and Generation

To evaluate the model we proposed, we collected 400 phishing sites from www.phishtank.com. We selected the phishing sites set from the repository of verified phishing sites provided by phishtank. The sites in the repository are submitted by users and then verified by voting. Recent studies have shown this repository to be mostly accurate, though vulnerable to attack [87].

In order to generate the feature vector from this site we have used our feature extractor algorithm so that the dataset is used for both training and testing.

Once the dataset is extracted for the purpose of training we have used 350 phishing and 300 legal webpages.

5.6 Experiment on Rule Based Components

A. URL Checker

In this component we have implemented a rule of thumb that tackles simple and common ways of attack that uses the manipulation of the URL. In this component we have generated a rule that considers different component of the URL.

The rules are based on the URL length, number of dots, the existence of suspicious special symbol, number of sensitive words, out of position domain name, port, the existence of redirected page, and using an IP in the URL.

Based on the result of each URL component we assigned 1 and 0 values for phishy and non-phishy behavior displayed by the webpage respectively. Once we collect the result of each URL component we calculated if 50% of the components results phishy. Even though we use these criteria as one aspect to classify the page as phishy we will not tolerate the suspicious symbol and IP rule violation. This is to maximize the detection accuracy because the two criteria will make up the majority of phishing Websites.

We have tasted this on four websites where three of them are taken form phishtank considered as phishy and the other is legal: http://confirmation-yours. The other websites are : infos.toysrfus.com/2f9a627242704a06778ecd377da1287e/, http://www.ebay.co.us-idtaqy2vag4nkvxxdhxbrtugfvag4nkvxxdhxbgsigl53347354.xn-42c6au4awb1a2c8i.com/56RETGE65RTGEY75464TERYR6534/index.html, http://cc4.co/KMQX and https://portal.aau.edu.et/.

The URL checker we developed has successfully classified the first two as phishy and the last URL as legitimate but it can't determine the class type of the third website which is phishy based on the phishtank report.

This is because the percentage value we find was less than 50% where the URL is found on only one criteria (port).

We have tested on majority of the URLs on phishtank that are classified as Phishy. We have found the accuracy to be around 70%. This is due to limited usage of URL manipulation in most phishing attacks. We used the following formula for calculating the accuracy:

$$Accuracy = Items\ Classified\ Correctly / All\ Items * 100\% \quad (1)$$

The URL checker has simplified the classification process since most phishing cites start manipulating the URL at most on the first place. So, in our proposed model we have started the detection process from this component that allows the user to get a report with a few minutes and without having a further damage. The checker will initiate the warning based on the user configuration but if this component is not able to decide the nature of the page it will transfer the flow of control to the machine learning based component.

B. Auto Fill Checker

We have designed and implemented an autofill checker that could work on most common browsers such as Mozilla Firefox, Chrome to detect a mismatch in auto fill features. We have used the difference in domain where the actual URL typed on the browser address bar and the submission by the auto fill feature. Besides the domain difference we have also checked the difference in the number of fields.

Due to the unavailability of data for testing, we have built a sample page that contains the attack by ourselves for validating the proposed method in detecting autofill based attack. As shown in Figure 5.1, the attack is to hide the username and password fields such that they are not visible on the webpage but are hidden and embedded in the html. It is built upon a form that gets inputs from user which is designed to get saved username and password.

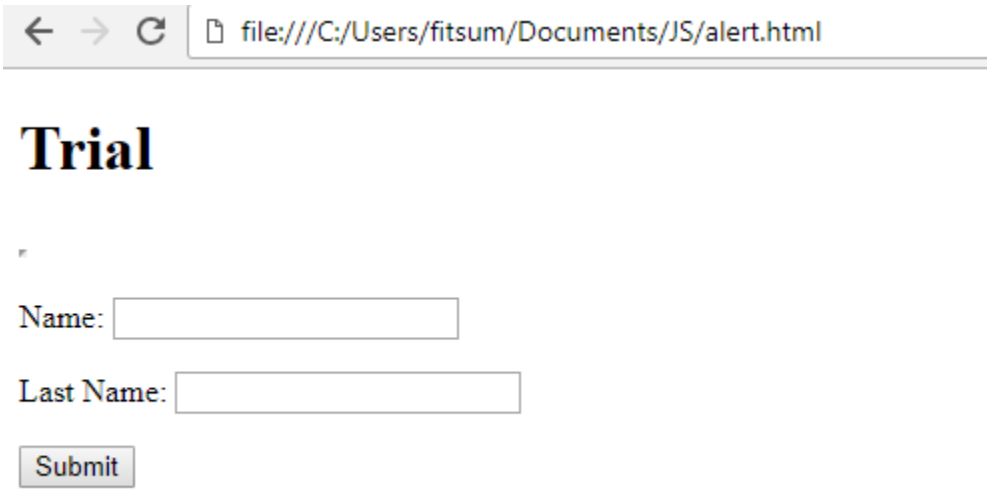

Figure 5.1: Sample Page for Autofill Attack

The submission of the form will send the values that are set whether they are hidden or not. The inputs that are filled by the browser such as username and password and also the action URL is spoofed.

As depicted in Figure 5.2, on submit, the URL is filled with fields that are not displayed on the form but are intentionally hidden. As seen on the link the parameters are:

- name: is set to the actual inputs that are supplied on the form
- lname: is also set from the input text field that are supplied on the form
- username: is set even if it is not displayed
- password: is also set to be sent to the malicious server even though it is hidden

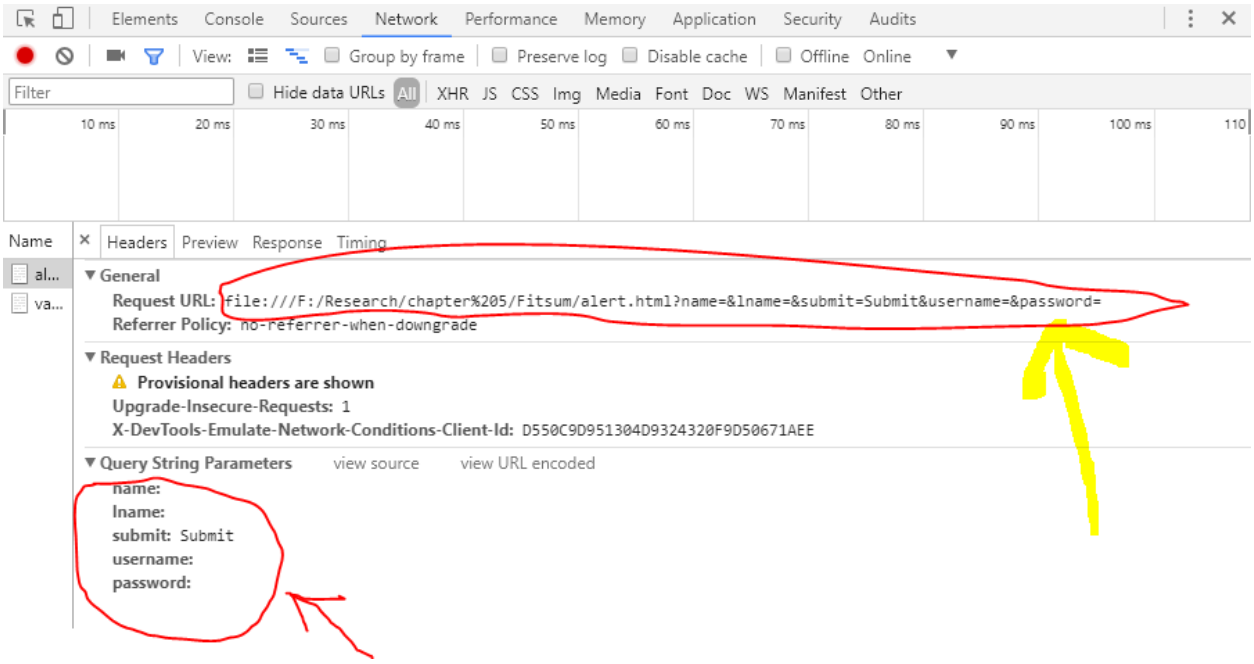

Figure 5.2: Data Transmission View on Console

The proposed solution starts out by checking if the action URL is the same as the link that displayed the form. There are two ways that this algorithm tries to decide if it is an attack or not.

- If the page URL matches the action URL then the fields that are in the form are checked if they are really displayed fully. If every field is not displayed the page is classified as a potential trait and the values of the hidden fields are set to none or an empty string
- If the page URL does not match the base URL of the document then the page is assumed to be an attack.

As shown in Figure 5.3, we have efficiently tracked the submission domain and number of fields and blocked if the feature finds a mismatch with sending null values.

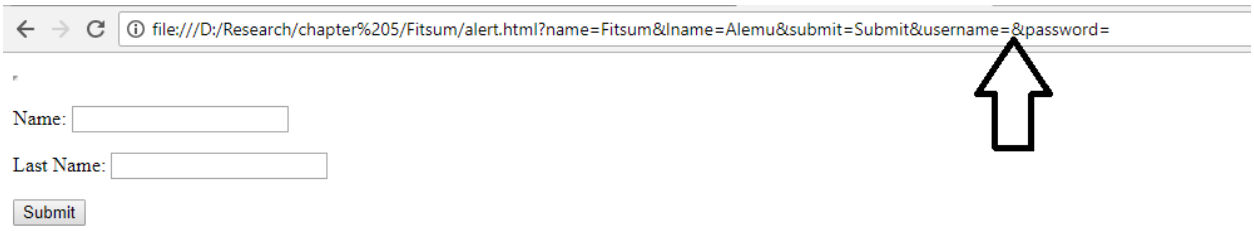

Figure 5.3: Sending Null Values

C. Auto Download Checker

This component tries to protect a system by trying to identify drive-by download attacks that are attached to an iframe tag that is embedded on the page which is linked to a malicious code that is not actually seen by the user.

In order to check the presence of a potential attack we pass through a sequence of steps. First, we create a page parser that downloads the page by accepting the URL of the page as a parameter. After downloading the page as a text, we go through the downloaded page trying to find the iframe tag.

The tag creates a box like structure when read by a browser (in a normal scenario) which is intentionally hidden by attackers to avoid suspicion. This is done by modifying the width and height attributes giving those values zero. Then we design a function that goes through all iframe tags inside the page and check if they are intentionally hidden from view. If that happens we tried to collects these tags.

Finally, we use a function that classifies the page as drive-by or not (i.e., True if it is or False if it is not) by checking if the collected potential iframes are non-empty.

Even though there might be many to be found as public derive by download websites, we have managed to demonstrate our proposed algorithm on few available. For example, we have tested on different webpages: https://www.hostinger.co.uk/error_403?static=true and http://askfmme.pe.hu/%D0%94%D0%BE%D0%B1%D1%80%D0%BE%20%D0%BF%D0%BE%D0%B6%D0%B0%D0%BB%D0%BE%0%B2%D0%B0%D1%82%D1%8C%20_%20%D0%92% D0% 9A%D0%BE%D0%BD%D1%82% D0%B0%D0%BA%D1%82%D0%B5.htm.

These two webpages have been reported as phishy in phishtank and we have tried to view them in different web browsers which then results in auto download of an application. The Auto download checker we proposed have successfully classified them as phishy and results in Iframe detected with the required specification and harmful software Installation message. We have also tested it on legitimate webpages which results in no Iframe detected.

The performance and accuracy of this component is highly dependent on the nature of the page where some pages might take different amount of time to download. Besides we have faced a little difficulty in fetching the pages due to Internet connection but we have tried to optimize the fetching

process. Besides the issues we have effectively send an interruption signal for stopping the download process from the client process.

5.7 Experiment on Mail Scanning and warning

A. Mail Scanner

We have tried to implement our own algorithm to extract URL for mail inbox. In order to execute the application, we need to manually allow if an application or device doesn't meet Google's security standards. Google will block anyone who tries to sign in from that application or device. Therefore, we need to manually allow access to less secure apps. It is illustrated on Figure 5.4.

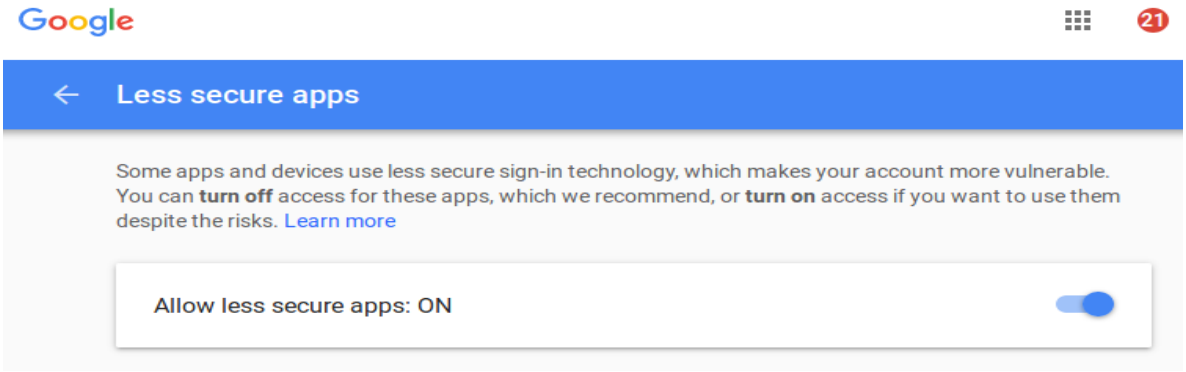

Figure 5.4: Security Configuration

Once allowed we have executed our algorithm to load mail box by providing a sample email address and password. For testing we have chosen unread message of Gmail specifically an inbox. Once the inbox object is loaded we have used a regular expression to extract a URL with either string or IP address. The result is shown on Figure 5.5.

```
Fast Debugger (ruby-debug-ide 0.6.1, debase 0.2.2, file filtering is supported) listens on 0.0.0.0:41861
=> Booting Puma
=> Rails 5.1.5 application starting in development
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.11.3 (ruby 2.5.0-p0), codename: Love Song
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://0.0.0.0:3000
Use Ctrl-C to stop
Started GET "/application" for 127.0.0.1 at 2018-03-21 14:54:03 +0300
Processing by ApplicationController#index as HTML

Started GET "/application" for 127.0.0.1 at 2018-03-21 15:09:46 +0300
Processing by ApplicationController#index as HTML
url found:
http
www.google.com
0> msg
=> [[["http", "www.google.com"]]]


0>
```

Figure 5.5: URL Extraction

As shown on Figure 5.5, the first part shows when the rails server starts to listen on port 300. Once the server started, it tries to send a request using GET for fetching the mail box. If the mail box is found, it tries to run the regular expression to find out URL on the body of a specific mailbox. Once it found the URL it returns a string message that contains:

url found : The protocol [*http*] The URL [*www.google.com*]

Our mail scanner has successfully extracted any URL that exists on the subject and body of the inbox. We have provided the extracted URL for classification in a JSON Format.

B. Mail Marker

Once the result of the classification is received as a JSON format that includes the phishing status and level of warning configuration, this component will apply different type of marker on mailbox. If the phishing status is phishy and the configuration is highlight inbox, the inbox message will be colored as shown in Figure 5.6.

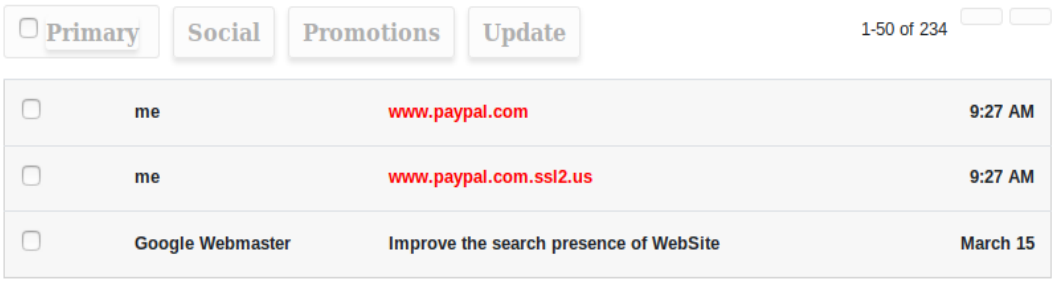

Figure 5.6: Highlight Inbox

This will be processed in the background as shown in Figure 5.7. The first part is as the usual rails servers start running on a specific port and making connection request. Once the mailbox is loaded successfully the specific mail in the inbox will be marked with a specific color.

```
Started GET "/gmail_phishings" for 127.0.0.1 at 2018-04-11 13:28:49 +0300
Processing by GmailPhishingsController#index as HTML
Message successfully marked as containing phishing url.
Message successfully marked as containing phishing url.
  Rendering gmail_phishings/index.html.erb within layouts/application
  Rendered gmail_phishings/index.html.erb within layouts/application (7.4ms)
Completed 200 OK in 10402ms (Views: 25.3ms)
```

Figure 5.7: Background Process on Marking

If the phishing status is phishy and the configuration is move to spam, the message will be moved to spam as shown in Figure 5.8.

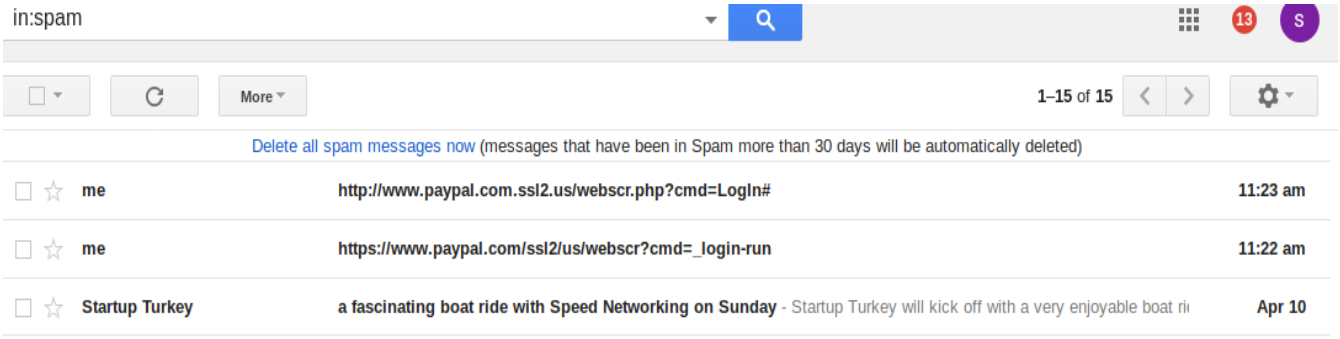

Figure 5.8: Moving to Spam

This will be processed in the background as shown in Figure 5.9. Once the rails servers start running on a specific port and making connection request, the mailbox will be loaded and a specific mail will be sent to spam.

```
Processing by GmailPhishingsController#index as HTML
Moving message to spam . . .
Message successfully moved to spam.
Moving message to spam . . .
Message successfully moved to spam.
  Rendering gmail_phishings/index.html.erb within layouts/application
  Rendered gmail_phishings/index.html.erb within layouts/application (148.7ms)
```

Figure 5.9: Background Process on Moving to Spam

If the phishing status is phishy and the configuration is remove inbox, the message will be removed form inbox shown in the background process in Figure 5.10.

```
Started GET "/gmail_phishings" for 127.0.0.1 at 2018-04-11 13:35:28 +0300
Processing by GmailPhishingsController#index as HTML
Message successfully deleted.
Message successfully deleted.
  Rendering gmail_phishings/index.html.erb within layouts/application
  Rendered gmail_phishings/index.html.erb within layouts/application (4.4ms)
Completed 200 OK in 6756ms (Views: 24.3ms)
```

Figure 5.10: Background Process on removing Inbox

5.8 EXPERIMENT ON MACHINE LEARNING BASED FEATURE

A) Experiment on our Identity Extractor

In order to start the training and testing we have designed our own algorithm to extract the page information that are considered as features. Once the features are extracted the value for each attribute is converted to feature vector. We have measured the accuracy of the feature vector by checking manually the pages and viewing the page information. We have found it very efficient and effective in retrieving all the features needed from both legitimate and phishy pages.

We have used the webpages that are reported and verified as phishy in phishthank but we have found that some features do not have appropriate values since most phishing websites try to manipulate specific features of the legal pages and some pages are currently offline. The page information extraction performance is shown in Table 5.2. The accuracy of the Identity Extractor is calculated based on *Equation 1*.

Table 5.2: Identity Extraction Performance

No	**Legitimate**	**Phishy**
Total pages	80	80
Correctly extracted	78	76
Accuracy	97.5%	95%

B) Experiment on the Page Classifier

We have used sklearn, a python API for implementing the SVM. Once the dataset is classified for testing and training we have started to train the SVM classifier. We have used dataset from phishtank and UCI with 350 phishing and 300 legal webpages.

We have used our feature vector extractor to build the data set for both the phishing and legal webpages. Once the features are extracted we run the firefly algorithm (FA) to find the optimized SVM parameters.

Various studies show that firefly algorithm is better in optimization problems, especially in parameter selection. In this algorithm there are two movement strategies used so far; Information based movement where we use the current firefly with the brighter values to create new firefly for

exploitation and Randomized movement where we use as exploration path. In randomized movement we use alpha (α) parameter that controls the randomness so it is recommend to reduce this value gradually.

In order to maximize the benefit of the algorithm we have to balance the exploitation and explorations. For the exploitation we have β where its values affect both α and γ (light absorption coefficient that tells how much of light is absorbed by air). If we increase gamma most of the light is absorbed by air then the fireflies will not be attracted to the brighter one so it moves randomly with higher search space. If we make it near zero the brightness will be the initial value where we will be trapped with local solutions rather than fining a global optimum solution for the parameter selection.

Parameters for the firefly algorithm were set empirically as follows: $\alpha = 0.2$, $\gamma = 0.23$ and $\beta_0 = 1$. Population size was 50, while the maximal number of iteration was set to 30. Additionally, if the accuracy did not improve in 10 consecutive iterations, the algorithm stops.

SVM parameters C and γ were searched in exponential space, search range for exponent for C was set to [-10; 20] and the SVM model was built for $C = 2^x$ where x is a parameter selected by FA, while exponent for parameter γ was searched in interval [-20;10] and similarly for SVM model $\gamma = 2^y$ was used where y was generated by FA.

Before we use the optimized parameters, we have tried on random values in order to test accuracy and efficiency. The accuracy of SVM with random values of parameters gives 85% where our proposed solution provides a much better accuracy which is 97%. This result shows that the classification accuracy has an excessive difference.

In order to prove quality of our proposed algorithm, we compared our results with other methods like SVM only, PSO and others. There are many ways to carry out the comparison of algorithm performance and two obvious approaches are: to compare the numbers of function evaluations for a given tolerance or accuracy, or to compare their accuracies for a fixed number of function evaluations.

C) Evaluation Metrics

Correct prediction (Accuracy): By comparing the classification predictions with the actual categories of the webpages, we are able to compute the numbers of true negatives (TN, correctly classified legal webpages), false negatives (FN, phishing page mistakenly classified as legal), true

positives (TP, correctly classified phishing page) and false positives (FP, legal pages mistakenly classified as phishing). To evaluate the classifier performance, we compute the accuracy with the following formula:

$$Accuracy = (TN+TP) / (TN+FP+TP+FN) \quad (2)$$

As shown in Figure 5.11, the proposed model predicts the phishing website more accurate than the existing methods.

Error Rate: Besides the correctness of the classification we are able to compute the misclassification rate by using the following formula:

$$Accuracy = (FP+FN) / (TN+FP+TP+FN) \quad (3)$$

As shown Figure 5.12, the proposed results in less error rate when compared with different optimization techniques like Bacteria Foraging algorithm, ACO and PSO with good number of iterations during optimization.

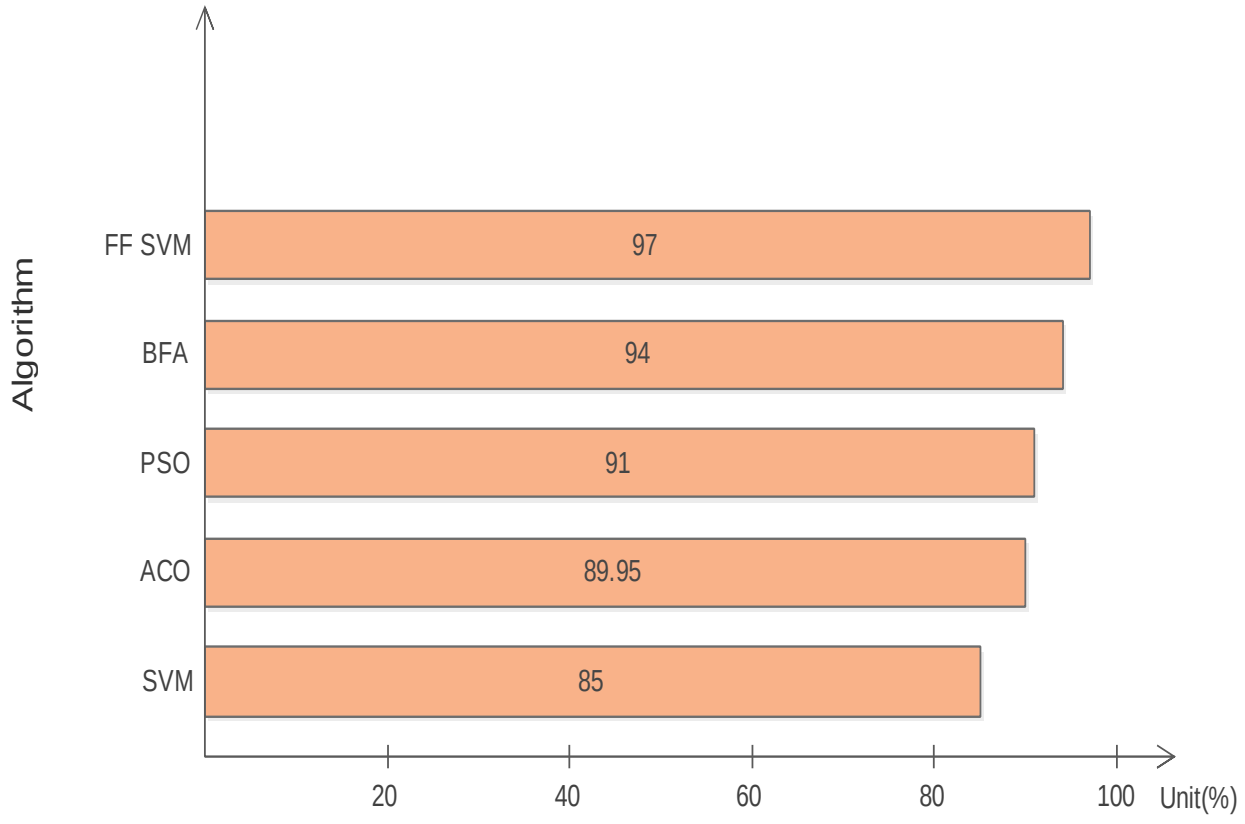

Figure 5.11: Comparison of Accuracy with Existing Algorithms

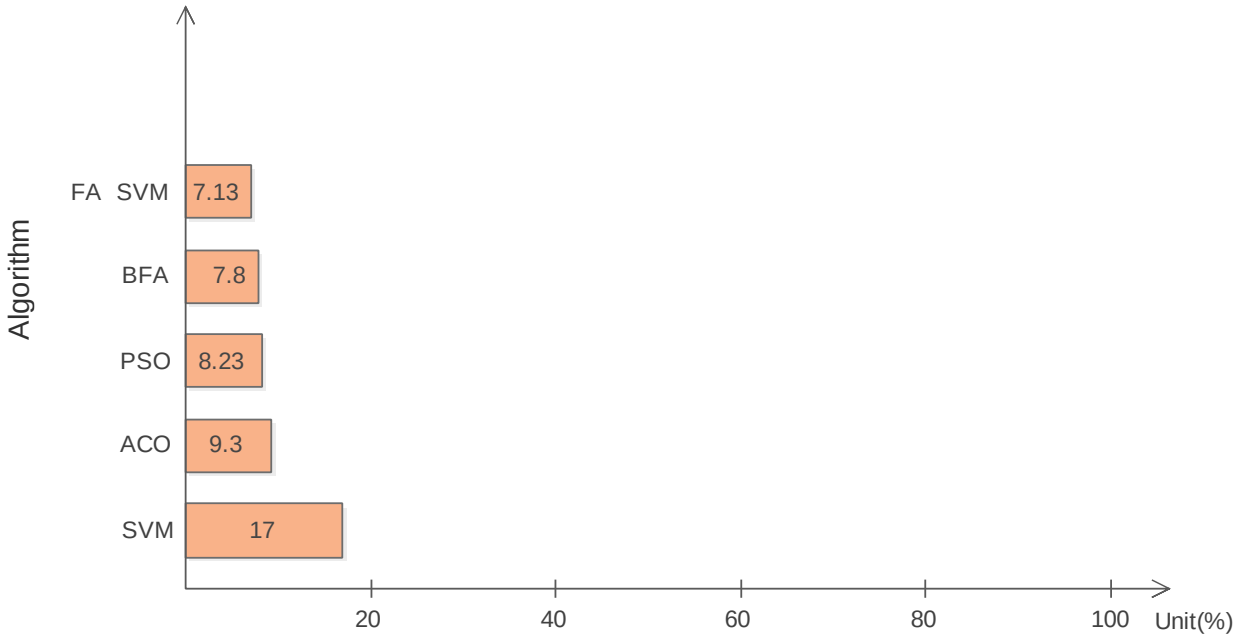

Figure 5.12: Comparison of Error rate with Existing Algorithms

5.9 PROTOTYPE

In order to test the majority of the components we have designed our own browser that has various functionalities. As shown on Figure 5.13, the user will type the URL of the web resource s/he wants to browse. Once the user starts browsing the phishing detection web service will run on the background as soon as a URL is typed on the address bar. Once the result comes from the web service, the user will be notified if phishing is detected.

The user is expected to configure initially the warning level that s/he will receive while visiting a webpage that comes from the classifier.

As shown in Figure 5.14, the user will need to provide configuration for the specific type warning in both the email and web visiting time.

The information the user provides are his/her full name for the reporting purpose, the reason why the user wants to set the warning (for general use or for testing), whether s/he wants to report phishing, specific warning during webpage visiting in browser, and email warning level.

Besides providing an initial configuration, the user can update previously configured values when s/he makes wrong configuration or not interested on his/her current configuration.

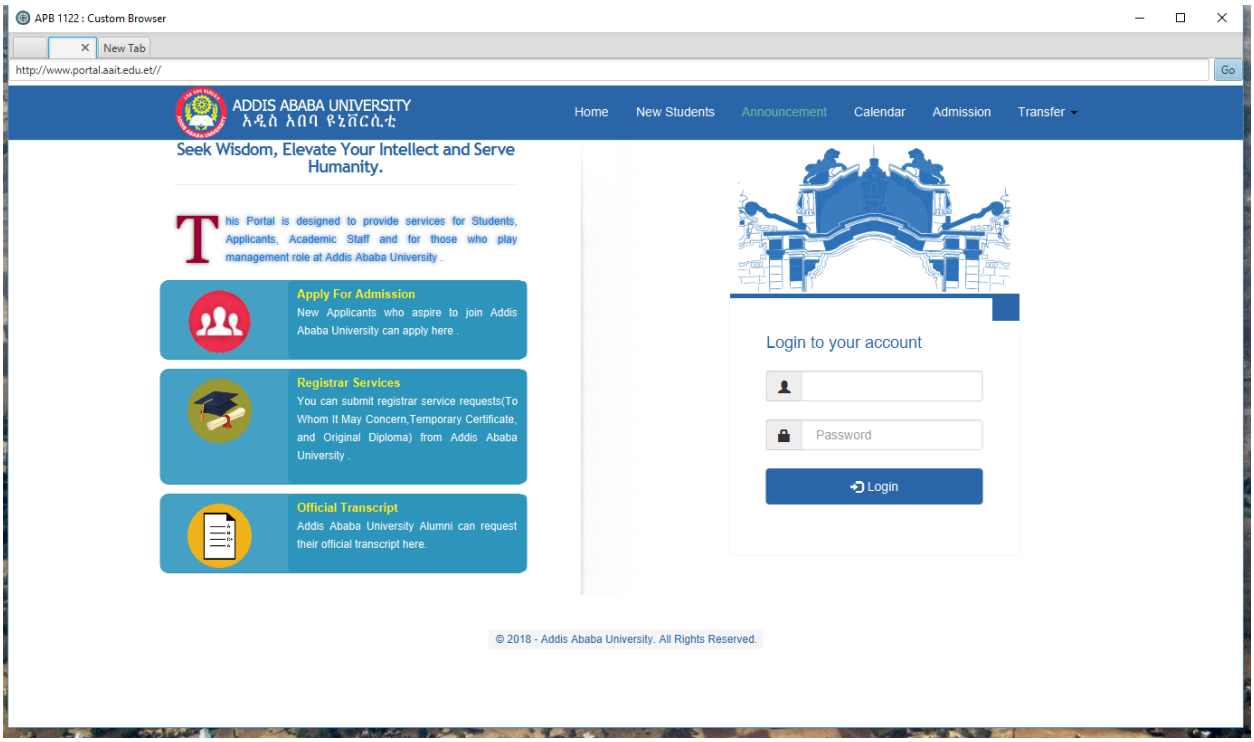

Figure 5.13: The Custom Browser

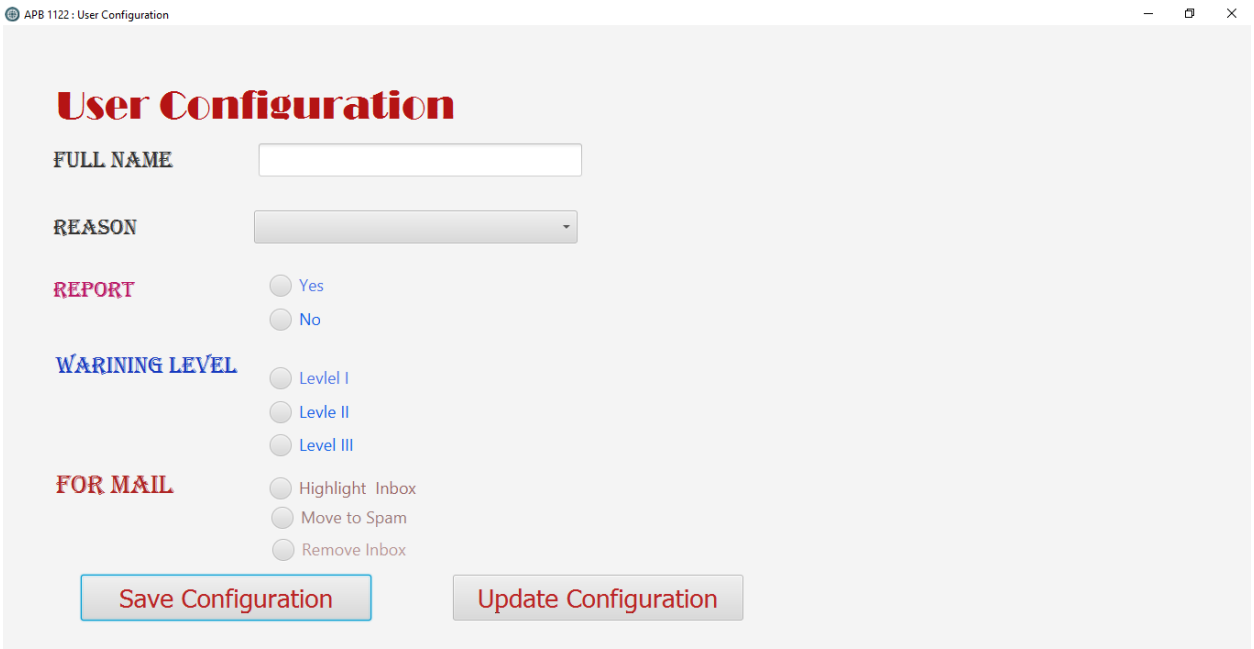

Figure 5.14: User Configuration

Once a URL is detected as a phisher page when the user starts to visit a page his/her browser warning configuration level will be verified. If the user configures level three s/he will be warned about the page as illustrated on Figure 5.15.

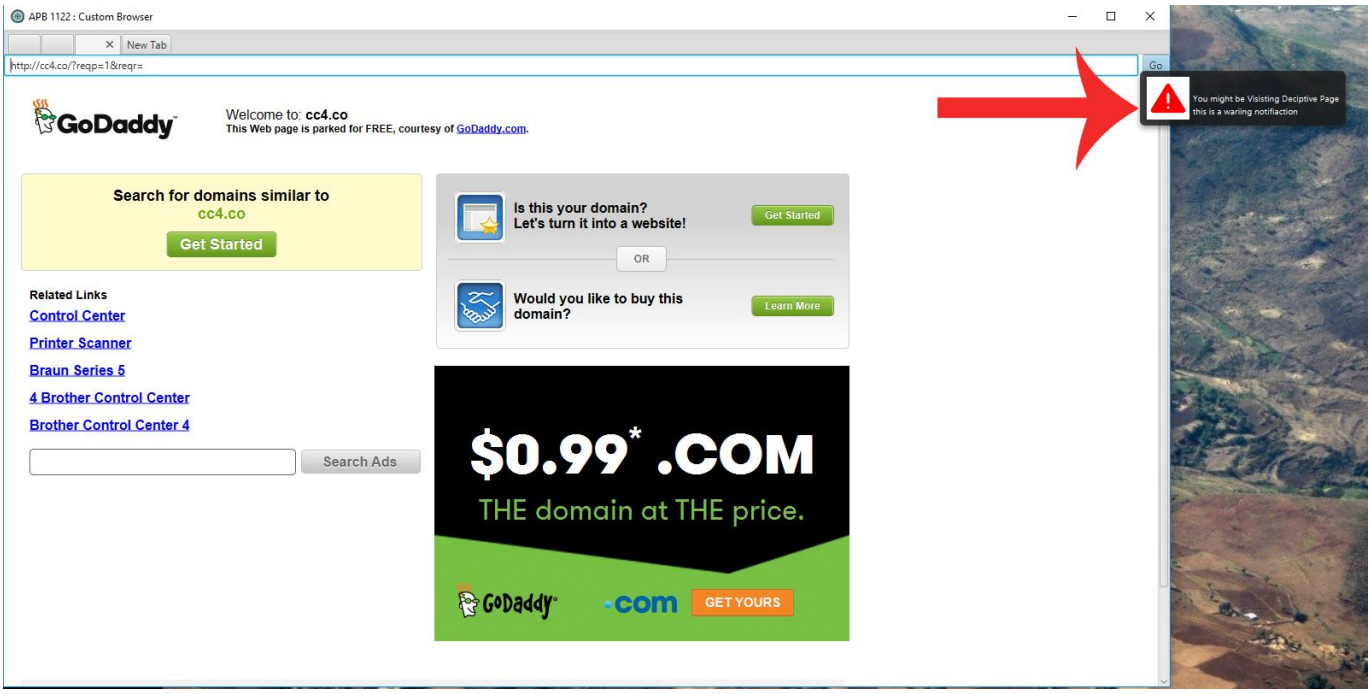

Figure 5.15: Level III User Warning

If the user has configured a warning level two s/he will receive a notification that does not allow the user to visit the page which is illustrated on Figure 5.16.

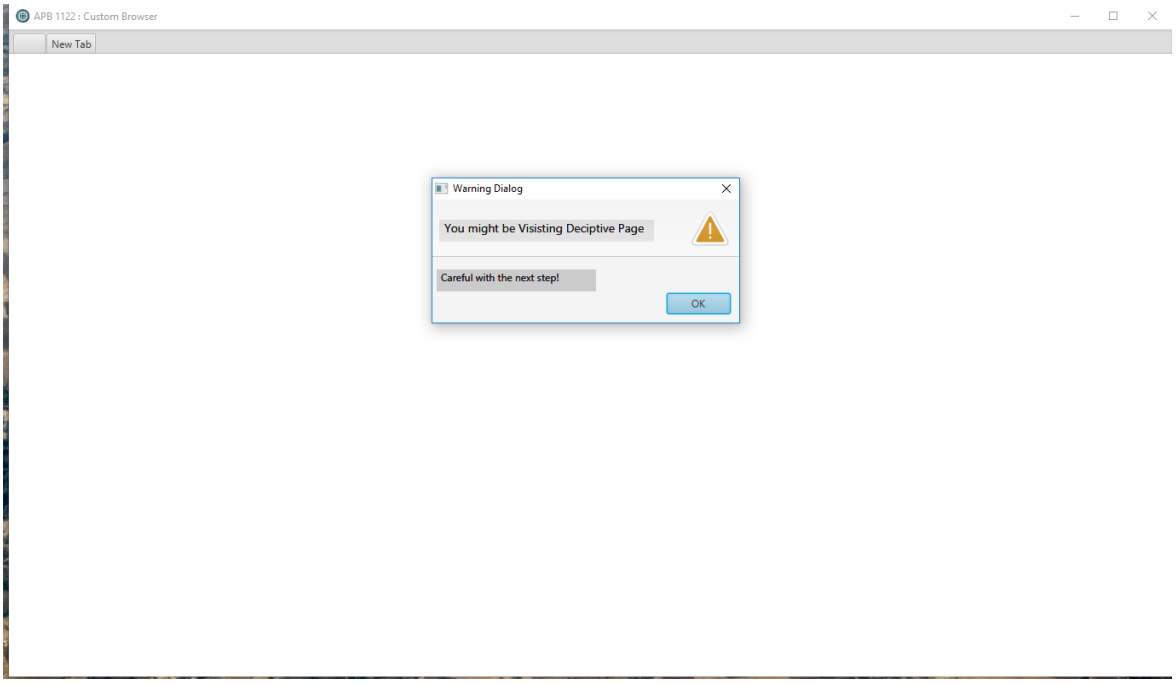

Figure 5.16: Level II Warning

On the web service side, we have developed a prototype for testing the feature extraction, feature generation and classification. In this model we have first used a URL from phishtank for the purpose of testing. Once the URL is typed on Browsers address bar the URL will be sent to the phishing detection web service. The URL extraction will be started and after that we have tested the classification and send the result to the client program with a JSON format. The prototype for this task is presented on Figure 5.17.

The figure also depicts the result of the URL feature extraction on the first text box. Besides, it also illustrates the result of the classification and the accuracy of the classifier on the second text box.

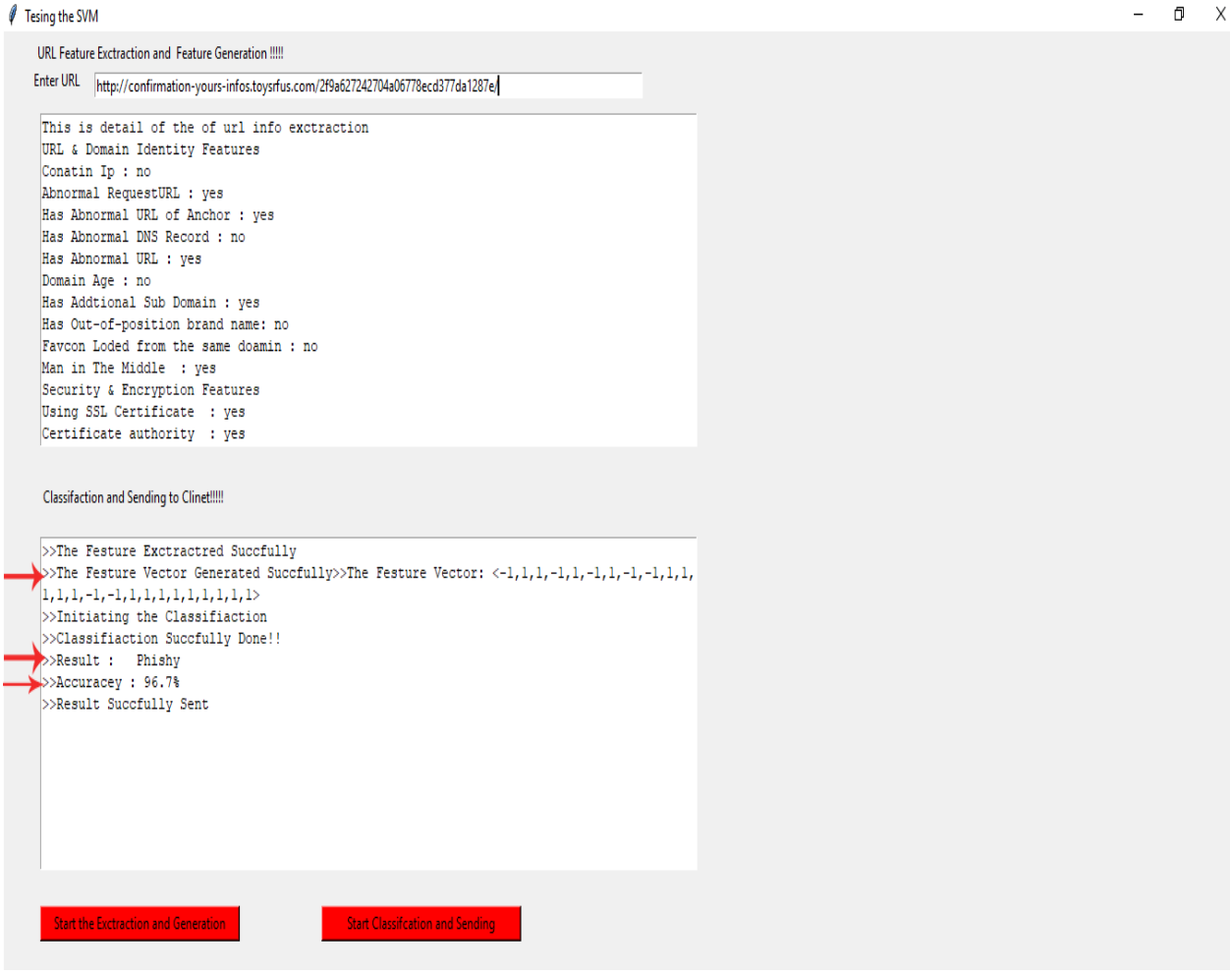

Figure 5.17: Backend Service

5. 10 ACCEPTANCE TESTING ON THE WARNING

We have tested the warning configuration and notification of both the mail and browser based with different types of volunteer individuals. As shown in Table 5.3, the result shows that it has better impact in understanding about phishing and sense of being protected. The users have also rated it with different scale but in general it has good acceptance. We have used rating out of five where five is high and one is low.

Table 5.3: Acceptance Scale of Warning

Frequency	*Rating*
7	4
5	5
4	3
8	4
Mode	4.5

Out of the sixteen evaluators nine of them are professionals and the rest seven are non-professionals. The professionals are programmers in the field of computer science and the others are those who use only computer. From the result obtained, we conclude that the warning method with full control of the user has better acceptance which is more than we expected.

5.11 DISCUSSION OF RESULT

Based on the outcomes observed in this Chapter we present here our observation of the result in relation to the research objectives stated in Chapter 1. As a reminder, the primary objective of this work was to design a model that enables detection of phishing websites and reporting them to user and answers the questions

- How to improve the predictive performance on classification?
- How to cope up with dynamic nature of phishing attack?
- How to generate a report and provide the information to the user appropriately?

Most phishing sites are simply copies of real sites. This property of phishing sites has made them difficult for humans to detect, but as we show, easy for computers. As we observed, the model computation time is a crucial aspect for any online anti-phishing approach, as users are unlikely to tolerate high performance penalties. So this was the major condition that was achieved through the

rule based elements even though we have observed a major improvement on the machine learning component.

Accordingly, the findings we have observed are encouraging considering the limited setting we have employed. To begin with, we have not implemented complex algorithms for many of the components within the framework (e.g., URL checker, Identity Extractor, Auto download checker and auto fill checker). However, even with the simple algorithms employed the prediction performance observed outperformed the initial assumption made.

As observed also in different research we have also observed that there is a significant relation between the two phishing website criteria; URL and Domain Identity and Security and Encryption for identifying phishing website. Also, we found insignificant trivial influence of the Page Style and Content.

Besides the rule-based elements our classifier has shown a major change in classification when the optimization algorithm is applied on parameter selection. The user management in the notification of information regarding the phishing report also shows a great acceptance based on the user testing we performed on volunteer users.

Chapter 6 : Conclusion and Future Works

6.1 CONCLUSION

It seems that it is not a common thing without hearing a news of a major corporation or public institution suffering a web security attack. From those different kinds of attacks phishing is one of the high-profile episodes that show how even organizations with ample resources struggle to fully protect their web properties and the data that lies behind them. What makes phishing a challenge for these organizations and even more so for smaller organizations with fewer resources at their disposal is that this threat is constantly evolving.

There have been different works that have been produced in order to tackle such problem but it is not still given a lot of attention of researchers because of its dynamic nature. The works that have been done follow two different approaches namely rule based and machine learning based approach. Both methods try to find useful patterns that could help in predicting the nature of the webpages where the user tries to access. The difference in these two methods is the way the page is classified as legal and phishy. In rule-based schemes, a page is classified as phishy or legal by generating some common rule and the user is provided a browser addon that depends on blacklist but on the case of machine learning approaches, intelligent algorithms learn these patters through experience (i.e., from the data set that they are trained).

Using only one approach is not applicable; Rule-based approach only depends on commonly known patterns and backlisting. On the other side machine learning approaches have gained a success in this area where patterns are discovered by machine learning algorithms for the purpose of classification. There have been a few works especially on the new trained nature inspired algorithms where they are used for the purpose of classification and optimization. These algorithms are mostly used for optimization problem in different areas where we have used firefly algorithm to optimize support vector machine classification parameters.

In this thesis, we have presented an approach to detect phishing attack from both browser access and email using link-based features. We have tried to use the hybrid approach by designing three rule-based elements and combing it with machine learning where we have used support vector machine learning algorithm optimized with firefly algorithm.

In the rule-based component we have proposed three different ways of tackling phishing attack mainly browser autofill, URL based and auto download feature. The contribution of the work mainly consists of the usage of features of links on user accessing browser, mail attachment and the hybrid nature of the framework with appropriate user notification. The proposed algorithm used in conjunction with the proposed prototype of web browser will help the user to get notified of possible phishing attacks and will prevent them from opening suspicious websites.

6.2 CONTRIBUTION

The primary contribution of this work is in developing a novel framework that could accommodate different kinds of phishing attacks by combing support vector machine with a multitude of nature inspired algorithms for optimization. Unlike other works, mentioned in the related work Chapter, we combine a modern nature inspired algorithm called firefly algorithm for optimizing the parameters of support vector machine. This has improved the accuracy of the classification when we compared it with other works done.

This work will also have a great contribution to the web security area by providing a baseline framework that can serve as one possible solution in the transition from contemporarily pure backlist-based browser technology towards the fully integrated phishing detection web service technology that incorporates both the rule based and machine learning approach.

This also helps the phishtank community to get more help from the majority of web users since the framework allows users to collaborate with this community and give a platform for the community to improve their service.

By including mail-based detection schema multiple mail service can adopt the framework for improving their service related to mail based phishing attacks. This improves user's application level experience that can help users to be protected from unwanted loss.

The integration of warning level module also contributes on user level experience on web users to have a better feeling of control and confidence. This will help users from losing different kinds of private information that are being sent to attackers.

The framework can also be a baseline in building a browser with a better security perimeter especially deceptive webpages that try to steal information by using the browser feature like autofill.

Even though our work is mainly pertinent to web phishing detection, since the framework's critical component is directed to user's webpage access prediction via collaboration, it can be adopted to other domains with similar requirements.

6.3 FUTURE WORKS

Due to time, computational resource and scope limitations we have not addressed a number of issues we believe important for the realization of the model we have proposed. One of the first issues is regarding the comparison of the current nature inspired algorithms we used on this thesis with other nature inspired algorithms and techniques in terms of their efficiency. In the future we planned to study further in detail how each nature inspired algorithm can be used in the optimization process of support vector machine algorithm.

Furthermore, we have planned to take a detail analysis of how to achieve a better continuous learning on support vector machine by designing and customizing the algorithm from scratch and evaluating it with additional dataset. Besides we would like to try to integrate the optimization in a best way that fits the learning process. The study also should thrive to include current and new machine learning approaches to tackle phishing attack that always changes forms of attack.

The other issue that is not covered in the thesis is the evaluation of the proposed work on the existing web browser. We have planned to provide plugin-based feature in order to address the issue.

The warning issue also needs to be evaluated by vast majority of individuals once we produce a plugin-based feature that can be open for user survey measure on the effectiveness of the warning that appears on the browser window and the labeling schema that we proposed for Gmail users.

The last issue that is not covered in this research is the identification of URL based phishing on other mail applications like Hotmail and Yahoo mail. In future we have planned to evaluate the effectiveness of identification of phishing URL on user inbox for other mail domain users. The evaluation we made on Gmail is due to the increase in users and the availability of free API for manipulating Gmail.

References

[1] Behdad, Mohammad, Luigi Barone, Mohammed Bennamoun, and Tim French. "Nature-inspired techniques in the context of fraud detection." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, no. 6 : 1273-1290, 2012.

[2] Dorigo, Marco, and Thomas Stützle. "The ant colony optimization metaheuristic: Algorithms, applications, and advances." In *Handbook of metaheuristics*, pp. 250-285. Springer, Boston, MA, 2003.

[3] Eiben, Agoston E., and James E. Smith. *Introduction to evolutionary computing*. Vol. 53. Heidelberg: springer, 2003.

[4] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: A survey," Contr. Eng. Pract., vol. 10, no. 11, pp. 1223–1241, 2002.

[5] I.Nikolos, K.Valavanis, N.Tsourveloudis, and A.Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," IEEETrans.Syst., Man, Cybern.B, Cybern., vol.33,no.6,pp.898–912, 2003.

[6] J. Wang ,X. Hong and R. Ren, T. Li, "A real-time intrusion detection system based on PSO-SVM", International Workshop on Information Security and Application (IWISA), Qingdao, China, pp. 319–321,2009.

[7] Aburrous, Maher, M. A. Hossain, Keshav Dahal, and Fadi Thabtah. "Associative classification techniques for predicting e-Banking phishing websites." In Multimedia Computing and Information Technology (MCIT), 2010 International Conference on, pp. 9-12. IEEE, 2010.

[8] Dhamija, Rachna, J. Doug Tygar, and Marti Hearst. "Why phishing works." In Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 581-590. ACM, 2006.

[9] Anti-Phishing Working Group. "Phishing attack trends report— First quarter" [Online]. Available: http: //www.antiphishing.org/ reports/apwg report Q1, 2010.

[10] John Wiley & Sons, The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition, 2011

[11] Webcohort, Inc. Only 10% of Web applications are secured against common hacking techniques. Http://www.imperva.com/ company/news/2004-feb-02.html, 2004.

[12] G. Hulme. New software may improve application security. http://www.informationweek.com/story/ IWK20010209S0003, 2001.

[13] Sense, Net. "Web-based security threats: how attacks have shifted and what to do about it GFI White Paper.", 2011.

[14] Jagatic, Tom N., Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. "Social phishing." Communications of the ACM 50, no. 10, 2007.

[15] Olson, David L., and Yanhong Li. "Mining fuzzy weighted association rules." In *System Sciences, 40th Annual Hawaii International Conference on*, pp. 53-53. IEEE, 2007.

[16] Lininger, Rachael, and Russell Dean Vines. *Phishing: Cutting the identity theft line*. John Wiley & Sons, 2005.

[17] Miller, Robert C., and Min Wu. "Fighting phishing at the user interface." *Security and Usability: Designing Secure Systems that People Can Use. O'Reilly*, 2005.

[18] Richmond, R. "Hackers set up attacks on home PCs, financial firms: study." *Retrieved September* 25 (2006): 2005.

[19] Wadlow, T. and Gorelik, V., 2009. Security in the Browser. *Queue*, *7*(2), pp.40-41, 2009.

[20] Wouter S. van Dongen, Browser security, 2009.

[21] Provos, Niels, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. "The Ghost in the Browser: Analysis of Web-based Malware.", HotBots 7 , 2007.

[22] Karthik Selvaraj and Nino Fred Gutierrez," The Rise of PDF Malware", 2010.

[23] Hillick, Mark. "Scareware traversing the world via a web app exploit." Bethesda (MD): SANS Institute InfoSec Reading Room, 2010.

[24] Gartner, INC. Gartner Says Number of Phishing E-Mails Sent to U.S. Adults Nearly Doubles in Just Two Years. Http://www.gartner.com/it/page.jsp?Id=498245, November 9 2006.

[25] Moore, Tyler, and Richard Clayton. "An Empirical Analysis of the Current State of Phishing Attack and Defence." In *WEIS*, 2007.

[26] Florencio, Dinei, and Cormac Herley. "A large-scale study of web password habits." In *Proceedings of the 16th international conference on World Wide Web*, pp. 657-666. ACM, 2007.

[27] Fogg, B. J., Jonathan Marshall, Othman Laraki, Alex Osipovich, Chris Varma, Nicholas Fang, Jyoti Paul et al. "What makes Websites credible? A report on a large quantitative study." In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 61-68. ACM, 2001.

[28] Downs, J. S., holbrook, M., and cranor, L. Decision Strategies and Susceptibility to Phishing. In Proceedings of the 2006 Symposium on Usable Privacy and Security, July 12-14, 2006.

[29] The statistics portal, https://www.statista.com/statistics/266161/websites-most-affected-by-phishing/, June 2017.

[30] See Larry Greenemeier, Update: AT&T Hackers Devised Elaborate Phishing Scam To Dupe Customers, Information Week, September 1, 2006.

[31] Afroz, Sadia, and Rachel Greenstadt. "Phishzoo: Detecting phishing websites by looking at them." In Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on, pp. 368-375. IEEE, 2011.

[32] Wilner, M., Bio-inspired and nanoscale integrated computing, Wiley, 2009.

[33] Yoshida, Zensho. *Nonlinear science: the challenge of complex systems*. Springer Science & Business Media, 2010.

[34] Jackson, Duncan E., and Francis LW Ratnieks. "Communication in ants." *Current biology* 16, no: R570-R574, 2006.

[35] Goss S, Aron S, Deneubourg J L, Pasteels J M. Self-organized shortcuts in the Argentine ant. Naturwissenschaften, , Vol 76, page 579–581, 1989.

[36] Dorigo M. Optimization, learning and natural algorithms. Ph.D.Thesis, Politecnico di Milano, Italy, 1992.

[37] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, no 1; 53–66, 1997.

[38] Baucer A, Bullnheimer B, Hartl R F, Strauss C. Minimizing total tardiness on a single machine using ant colony optimization. Central European Journal for Operations Research, 8, 125–141, 2000.

[39] Toth P, Vigo D. Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Applied Mathematics, 123, 487–512, 2002.

[40] Secomandi, Nicola. "Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands." *Computers & Operations Research* 27, no. 11-12: 1201-1225, 2000.

[41] Aardal K I, van Hoesel S P M, Koster A M C A, Mannino C, Sassano A. Models and solution techniques for the frequency assignment problem. A Quarterly Journal of Operations Research, no1;261–317,2001.

[42] Pham D T, Ghanbarzadeh A, Koc E, Otri S, Rahim S, Zaidi M. The Bees Algorithm, Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.

[43] Seeley T D. The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies, Harvard University Press, Cambridge, 1996.

[44] Frisch K V. Bees: Their Vision, Chemical Senses and Language, revised edition, Cornell University Press, Ithaca, New York, 1976.

[45] Pham, Duc Truong, Afshin Ghanbarzadeh, Ebubekir Koç, Sameh Otri, S. Rahim, and Muhamad Zaidi. "-The Bees Algorithm—A Novel Tool for Complex Optimisation Problems." In *Intelligent Production Machines and Systems*, pp. 454-459, 2006.

[46] Pham, D. T., A. Ghanbarzadeh, E. Koc, and S. Otri. "Application of the bees algorithm to the training of radial basis function networks for control chart pattern recognition." In *Proceedings of 5th CIRP international seminar on intelligent computation in manufacturing engineering (CIRP ICME'06), Ischia, Italy*, pp. 711-716, 2006.

[47] Olague, Gustavo, and Cesar Puente. "Honeybees as an Intelligent based Approach for 3D Reconstruction." In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1, pp. 1116-1119. IEEE, 2006.

[48] Barricelli, Nils Aall. "Symbiogenetic evolution processes realized by artificial methods." *Methodos* 9, no. 35-36: 143-182, 1957.

[49] Fraser, Alex, and Donald Burnell. "Computer models in genetics." *Computer models in genetics*, 1970.

[50] Cha, Sung-Hyuk, and Charles C. Tappert. "A genetic algorithm for constructing compact binary decision trees." *Journal of pattern recognition research* 4, no. 1: 1-13, 2009.

[51] Ting, Chuan-Kang. "On the mean convergence time of multi-parent genetic algorithms without selection." In *European Conference on Artificial Life*, pp. 403-412. Springer, Berlin, Heidelberg, 2005.

[52] Zang, H., Zhang, S. and Hapeshi, K., 2010. A review of nature-inspired algorithms. *Journal of Bionic Engineering, 7*, pp.S232-S237, 2010.

[53] Gondro, Cedric, and Brian P. Kinghorn. "A simple genetic algorithm for multiple sequence alignment." *Genetics and Molecular Research* 6, no. 4: 964-982, 2007.

[54] Hill, Tobias, Andor Lundgren, Robert Fredriksson, and Helgi B. Schiöth. "Genetic algorithm for large-scale maximum parsimony phylogenetic analysis of proteins." *Biochimica et Biophysica Acta (BBA)-General Subjects* 1725, no. 1: 19-29, 2005.

[55] Li, Yun, Kiam Heong Ang, Gregory CY Chong, Wenyuan Feng, Kay Chen Tan, and Hiroshi Kashiwagi. "CAutoCSD-evolutionary search and optimisation enabled computer automated control system design." *International Journal of Automation and Computing* 1, no. 1: 76-88, 2004.

[56] Babu, B. Gajendra, and M. Kannan. "Lightning bugs." *Resonance* 7, no. 9: 49-55, 2002.

[57] Yang, X.S. and Algorithms, N.I.M., Luniver Press. *Beckington, UK*, pp.242-246, 2008.

[58] Yang, Xin-She. "Firefly algorithms for multimodal optimization." In *International symposium on stochastic algorithms*, pp. 169-178. Springer, Berlin, Heidelberg, 2009.

[59] Eberhart, Russell, and James Kennedy. "A new optimizer using particle swarm theory." In *Micro Machine and Human Science, 1995. MHS'95, Proceedings of the Sixth International Symposium on*, pp. 39-43. IEEE, 1995.

[60] T. Baeck, "Generalized convergence models for tournament and (mu,lambda)-selection." Proc. of the Sixth International Con$ on Genetic Algorithms, pp. 2-7, Morgan Kaufmann Publishers, San Francisco, CA, 1995.

[61] Eberhart, Russell, and James Kennedy. "A new optimizer using particle swarm theory." In *Micro Machine and Human Science, 1995. MHS'95, Proceedings of the Sixth International Symposium on*, pp. 39-43. IEEE, 1995.

[62] Poli, Riccardo, James Kennedy, and Tim Blackwell. "Particle swarm optimization." *Swarm intelligence* 1, no. 1: 33-57, 2007.

[63] Yao, Xin. "Evolving artificial neural networks." *Proceedings of the IEEE* 87, no. 9: 1423-1447, 1999.

[64] Parsopoulos, K. El, V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis. "Objective function" stretching" to alleviate convergence to local minima." *Nonlinear Analysis-Theory Methods and Applications* 47, no. 5: 3419-3424, 2001.

[65] Parsopoulos, K. E., V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis. "Stretching technique for obtaining global minimizers through particle swarm optimization." In *Proceedings of the Particle Swarm Optimization Workshop*, vol. 29. Indianapolis, USA, 2001.

[66] Enache, Adriana-Cristina, and Victor Valeriu Patriciu. "Intrusions detection based on support vector machine optimized with swarm intelligence." In Applied Computational Intelligence and Informatics (SACI), 2014 IEEE 9th International Symposium on, pp. 153-158. IEEE, 2014.

[67] Analytics Vidhya, https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/, 2017

[68] P. Marrow, "Nature-inspired computing technology and applications," BT Technol. J., vol. 18, pp. 13–23, 2000.

[69] Mohammad, Rami M., Fadi Thabtah, and Lee McCluskey. "Predicting phishing websites using neural network trained with back-propagation." In Proceedings on the International Conference on Artificial Intelligence (ICAI), p. 1. The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2013.

[70] Radha Damodaram, M. C. A., and M. L. Valarmathi. "Bacterial Foraging Optimization for Fake Website Detection." International Journal of Computer Science & Applications (TIJCSA) 1, no. 11, 2013.

[71] E. Kirda and C. Kruegel, "Protecting users against phishing attacks," The Computer Journal, 2005.

[72] Jain, Aanchal, and Vineet Richariya. "Implementing a web browser with phishing detection techniques." arXiv preprint arXiv: 1110.0360, 2011.

[73] Y. Zhang, J. Hong and L. Cranor, "CANTINA: A ContentBased Approach to Detect Phishing Websites," in Proceedings of the 16th World Wide Web Conference, Banff, Alberta, Canada, 2007.

[74] M. Aburrous, M. A. Hossain, K. Dahal and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," Expert Systems with Applications: An International Journal, pp. 7913-7921, December 2010.

[75] Y. Pan and X. Ding, "Anomaly Based Web Phishing Page Detection," in In ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference., Washington, DC, Dec. 2006.

[76] Egelman, Serge, Lorrie Faith Cranor, and Jason Hong. "You've been warned: an empirical study of the effectiveness of web browser phishing warnings." In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1065-1074. ACM, 2008.

[77] He, Mingxing, Shi-Jinn Horng, Pingzhi Fan, Muhammad Khurram Khan, Ray-Shine Run, Jui-Lin Lai, Rong-Jian Chen, and Adi Sutanto. "An efficient phishing webpage detector." *Expert systems with applications* 38, no. 10 : 12018-12027, 2011.

[78] Mohammad, Rami M., Fadi Thabtah, and Lee McCluskey. "Intelligent rule-based phishing websites classification." *IET Information Security* 8, no. 3: 153-160, 2014.

[79] Garera, Sujata, Niels Provos, Monica Chew, and Aviel D. Rubin. "A framework for detection and measurement of phishing attacks." In Proceedings of the 2007 ACM workshop on Recurring malcode, pp. 1-8. ACM, 2007.

[80] lifehacker, https://lifehacker.com/your-browsers-autofill-data-can-be-phished-heres-how-t-1791084371, December 2018.

[81] Cova, Marco, Christopher Kruegel, and Giovanni Vigna. "Detection and analysis of drive-by-download attacks and malicious JavaScript code." In Proceedings of the 19th international conference on World Wide Web, pp. 281-290. ACM, 2010.

[82] How-To Geek, https://www.howtogeek.com/119458/htg-explains-whats-a-browser-cookie/, December 2018

[83] Yang, Xin-She. "Firefly algorithm, stochastic test functions and design optimisation." *International Journal of Bio-Inspired Computation* 2, no. 2: 78-84, 2010.

[84] Eva Tuba, Milan Tuba, and Dana Simian. Adjusted bat algorithm for tuning of support vector machine parameters. In IEEE Congress on Evolutionary Computation (CEC), pages 2225–2232. IEEE, 2016.

[85] Eva Tuba, Milan Tuba, and Marko Beko. Support vector machine parameters optimization by enhanced fireworks algorithm. In Ying Tan, Yuhui Shi, and Ben Niu,editors, LNCS, Advances in Swarm Intelligence: 7th International Conference on Swarm Intelligence, Proceedings, Part I, volume 9712, pages 526–534. Springer International Publishing, 2016.

[86] RESTful Web Services for Service Provisioning in Next-Generation Networks: A Survey. IEEE Communications Magazine. December 2011

[87] Moore, Tyler, and Richard Clayton. "Evaluating the wisdom of crowds in assessing phishing websites." In International Conference on Financial Cryptography and Data Security, pp. 16-30. Springer, Berlin, Heidelberg, 2008.

Declaration

I, Undersigned, declare that this thesis is my original work and has not been presented for degree in any other university, and that all sources of material used for the thesis have been acknowledged.

Declared by:

Name: **Fitsum Alemu Degfu**

Signature: _____

Date _____

Confirmed by advisor:

Name: **Mulugeta Libsie (PhD)**

Signature: _____

Date _____

Place and date of submission: Addis Ababa University, June 15, 2018.