CHAPTER 1

INTRODUCTION

1.1 PHISHING

Phishing is a new word produced from 'fishing', it refers to the act that the attacker allure users to visit a faked Web site by sending them faked e-mails (or instant messages), and stealthily get victim's personal information such as user name, password, and national security ID, etc. This information then can be used for future target advertisements or even identity theft attacks (e.g., transfer money from victims' bank account).

The frequently used attack method is to send e-mails to potential victims, which seemed to be sent by banks, online organizations, or ISPs. In these e-mails, they will make up some causes, e.g. the password of your credit card had been mis-entered for many times, or they are providing upgrading services, to allure you visit their Web site to conform or modify your account number and password through the hyperlink provided in the e-mail. If you input the account number and password, the attackers then successfully collect the information at the server side, and is able to perform their next step actions with that information (e.g., withdraw money out from your account).Phishing itself is not a new concept, but it's increasingly used by phishers to steal user information and perform business crime in recent years. Within one to two years, the number of phishing attacks increased dramatically.

1.2 SYSTEM OVERVIEW

The common characteristics of the hyperlinks in phishing e-mails. Our analysis identifies that the phishing hyperlinks share one or more characteristics as listed below. The visual link and the actual link are not the same; The attackers often use dotted decimal IP address instead of DNS name. Special tricks are used to encode the hyperlinks maliciously. The attackers often use fake DNS names that are similar (but not identical) with the target website.

1.2 SCOPE OF THE PROJECT

An end-host based anti-phishing algorithm which we call Link Guard, based on the characteristics of the phishing hyperlink. Since Link Guard is character-based, it can detect and prevent not only known phishing attacks but also unknown ones. We have implemented Link Guard in Windows, and our experiments indicate that Link Guard is light-weighted in that it consumes very little memory and CPU circles, and most importantly, it is very effective in detecting phishing attacks with minimal false negatives.

CHAPTER 2

LITERATURE SURVEY

[1]Akhilendra Pratab Singh and Vimal Kumar, "Detection and Prevention of Phishing Attack Using Dynamic Watermarking", International Conference on Advances in Information Technology and Mobile Communication, 2017.

This paper proposes a novel anti-phishing approach based on Dynamic watermarking technique. According to this approach user will be asked for some additional information like watermark image, its fixing position and secret key at the time of user's registration and these credentials of particular user will be changed at per login. During each login phase a user will verify the authentic watermark with its position and decide the legitimacy of website. There is some limitation in this approach like many checks and enforcements which are used by the client-side defense tools can be tricked by attackers after getting a reasonable knowing of web site construction. For example, using mosaic attack, an attacker can make fool the image check system of Spoof Guard by partitioning the logo image into small parts and show it in such way that it looks like legitimate one. Sometime due to irritation of some protection means like antivirus, user turnoff the protection mechanism of client side.

[2]Sadia Afroz and Rachel Greenstadt, "PhishZoo: Detecting Phishing Websites by Looking at Them", Fifth IEEE International Conference on Semantic Computing (IEEEICSC), 2011.

This paper proposes a phishing detection approach PhishZoo that uses profiles of trusted websites' appearances to detect phishing. This approach provides similar accuracy to blacklisting approaches (96%), with the advantage that it can classify zero-day phishing attacks and targeted attacks against smaller sites (such as corporate intranets). A key contribution of this paper is that it includes a performance analysis and a framework for making use of computer vision techniques in a practical way.

The limitation is that this approach depends only on websites content to detect corresponding phishing sites. It can detect new phishing sites which are not yet blacklisted and targeted attacks against small brokerages and corporate intranets.

Another limitation is that PhishZoo's current image matching algorithm fails to find matching in cases where the logos are rotated (more than 30 degrees), combined with other site elements or cropped.

[3]V. Shreeram and M. Suban, “Anti-phishing detection of phishing attacks using genetic algorithm”, International Conference on Communication Control and Computing Technologies, 2010.

In this paper, the rule based system formed by genetic algorithm is proposed, which can be utilized as a part of an enterprise solution to anti-phishing. A legitimate webpage owner can use this approach to search the web for suspicious hyperlinks. Genetic algorithm is used to evolve rules that are used to differentiate phishing link from legitimate link. The main idea proposed is that evaluating the parameters like evaluation function, crossover and mutation, GA generates a ruleset that matches only the phishing links. This ruleset is stored in a database and a link is reported as a phishing link if it matches any of the rules in the rule based system and thus it keeps safe from fake hackers.

The limitation is that in order to fully exploit the suspicious level, It needs to examine all fields related with a specific URL in Phishing e-mail and also there are many parameters to consider for the application of GA.

[4]Vimal Kumar and Rakesh Kumar, ”Detection of phishing attack using visual cryptography in ad hoc network”, International Conference on Communications and Signal Processing, 2015.

In this paper, a novel anti-phishing approach is proposed based on visual cryptography. According to this approach a user generates two shares of an image using (2, 2) visual cryptography scheme. Client stores the first share of this image and second share is uploaded to the website at the time of user registration. After this, website asks for some other information like second share of the image, user name, and password. These credentials of a particular user can change once per login. During each login phase, a user verifies the legitimacy of a website by getting secret information with the help of stacking both shares. This does not suffer from False Positive (FP) notification.

The disadvantage is that for every time the user has to login, It will needs to verify the legitimacy of website by getting secret information with the help of stacking both shares. And also it is image based, So that images also can be cleverly spoof by attackers.

[5]Weiwei Zhuang, Qingshan Jiang and Tengke Xiong, "An Intelligent Anti-phishing Strategy Model for Phishing Website Detection", International Conference on Distributed Computing Systems Workshops, 2012.

An intelligent anti-phishing strategy model for phishing website detection and categorization through learning and training samples from large and real daily phishing websites collected form Kingsoft Internet Security Lab. They first parse and analyze the webpage content and extract 10 different types of features such as title, keywords, description, alt and link text information to represent the webpage. Then we build heterogeneous classifiers according to the characteristics of different features. Finally, an ensemble method is used to combine the prediction results of these heterogeneous classifiers for phishing detection, and a hierarchical clustering algorithm is employed for categorizing the phishing websites. Experiments on real life datasets demonstrate that our method outperforms existing popular detection methods and commonly used anti-phishing tools in phishing detection.

This solution mainly makes the use of the characteristics of the URL that includes URL similarity calculation, domain name probability evaluation, number of the external links of the webpage, IP address, the port number, etc.

[6]Abdelhamid N, Ayesh A and Thabtah F, "Phishing Detection Based on Associative Classification Data Mining", 2014.

The problem of phishing detection is investigated using AC approach in data mining. We primarily test a developed AC algorithm called MCAC and compare it with other AC and rule induction algorithms on phishing data. The phishing data have been collected from the Phish tank archive (Phish Tank, 2006), which is a free community site. In contrast, the legitimate websites were collected from yahoo directory. The evaluation measures used in the comparison are accuracy, number of rules, any label, and label-weight (Thabtah, Cowling, & Peng, 2004).

This approach normally devises classifiers (set of rules) that are simple yet accurate. The decision-making process becomes reliable because these decisions are made based on rules discovered from historical data by the AC algorithm. Although plenty of applications are available for combating phishing websites few of them make use of AC data mining.

[7]Sun Bin,Wen, Qiaoyan and Liang Xiaoying, "A DNS Based Anti-phishing Approach, Second International Conference on Networks Security, Wireless Communications and Trusted Computing, 2010.

It presents the design and implementation of a DNS based anti-phishing approach, which can be used to protect the card number and the password of the online bank users effectively, and prevent phishers and pharmers from stealing such information. First, the bank name, its DNS server's IP address, and the card number range will be stored in the database. If the Phishing Detecting Device detects that a bank card number is being sent to a suspicious website, the device will send an inverse DNS query to the DNS server of the related bank. By verifying whether the suspicious website is within the domain of the bank, it can determine whether the website is a phishing website.

It can be used to protect the account with some special characteristic such as e-mail account. Because in the e-mail address, after the symbol "@" there is a domain name of the e-mail service provider, it is convenient for us to obtain the DNS address of the e-mail service provider, and we can use our proposed approach to protect email users to login the mailbox safely through web.

[8]Jayshree Hajgude and Lata Ragha, "Phish mail guard: Phishing mail detection technique by using textual and URL analysis", World Congress on Information and Communication Technologies, 2012.

It proposes a technique where they consider the advantages of blacklist, white list and heuristic technique for increasing accuracy and reducing false positive rate. In heuristic technique we are using textual analysis and URL analysis of e-mail. Since most of the phishing mails have similar contents, their proposed method will increase the performance by analyzing textual contents of mail and lexical URL analysis. It will detect phishing mail if DNS in actual link is present in blacklist. DNS is present in white list then it is considered as legitimate DNS. If it is not present in blacklist as well as white list then it is analyzed by using pattern matching with existing phishing DNS, contents found in mail and analysis of actual URL. With the help blacklist and white list we are avoiding detection time for phishing and legitimate email. At the same time we are decreasing false positive rate by combining features of DNS, textual content analysis of email and URL analysis.

However, phishing has become more and more complicated and sophisticated so that phishers can bypass the filter set by current anti-phishing techniques and cast their bait to customers and organizations.

[9]Routhu Srinivasa Rao and Syed Taqi Ali, "A Computer Vision Technique to Detect Phishing Attacks", Fifth International Conference on Communication Systems and Network Technologies, 2015.

The proposed approach is a combination of whitelist and visual similarity based techniques. We use computer vision technique called SURF detector to extract discriminative key point features from both suspicious and targeted websites. Then they are used for computing similarity degree between the legitimate and suspicious pages. Our proposed solution is efficient, covers a wide range of websites phishing attacks and results in less false positive rate. The basic idea is to maintain a legitimate image database consisting of all popular website screenshots along with their URLs (whitelist), Obtain the accessed URL and do comparing with whitelist of URLs, If comparison is successful URL will be considered as Innocent and no further checking will be required and This removes the extra overhead of comparison of legitimate website display.

And also Extract the SURF features from both suspicious website screenshot, image database and compare for similarity check. If similarity score is greater than the threshold, webpage is considered as suspected. If similarity score is less than threshold, URL is considered as innocent. The domain will be the part of white-list in next update.

[10]Chuan Yue and Haining Wang, "Anti-Phishing in Offense and Defense", Annual Computer Security Application Conference, 2008.

It proposes a new approach to protect against phishing attacks with “bogus bites”. They develop Bogus Biter, a unique client-side anti phishing tool, which transparently feeds a relatively large number of bogus credentials into a suspected phishing site. Bogus Biter conceals a victim’s real credential among bogus credentials, and moreover, it enables a legitimate web site to identify stolen credentials in a timely manner. Leveraging the power of client-side automatic phishing detection techniques, Bogus Biter is complementary to existing preventive anti-phishing approaches. They implement Bogus Biter as an extension to Firefox 2 web browser, and evaluate its efficacy through real experiments on both phishing and legitimate web sites.

They endeavor to prevent users from being tricked into revealing their credentials to phishing sites. Nevertheless, these prevention-based approaches alone are insufficient to shield vulnerable users from “biting the bait” and defeat phishers, as human users are the weakest link in the security chain. The ever-increasing prevalence and severity of phishing attacks clearly indicate that anti-phishing is still a daunting challenge.

CHAPTER 3

SYSTEM ANALYSIS

3.1EXISTING SYSTEM

3.1.1 Detect and block the phishing Web sites in time:

If we can detect the phishing Web sites in time, we then can block the sites and prevent phishing attacks. It's relatively easy to (manually) determine whether a site is a phishing site or not, but it's difficult to find those phishing sites out in time. Here we list two methods for phishing site detection.

A) The Web master of a legal Web site periodically scans the root DNS for suspicious sites (e.g.www.1cbc.com.cnvs.www.icbc.com.cn).

B) Since the phisher must duplicate the content of the target site, he must use tools to (automatically) download the Web pages from the target site.

It is therefore possible to detect this kind of download at the Web server and trace back to the phisher. Both approaches have shortcomings. For DNS scanning, it increases the overhead of the DNS systems and may cause problem for normal DNS queries, and furthermore, many phishing attacks simply do not require a DNS name. For phishing download detection, clever phishers may easily write tools which can mimic the behavior of human beings to defeat the detection.

3.1.2 Enhance the security of the web sites:

The business Websites such as the Web sites of banks can take new methods to guarantee the security of users' personal information. One method to enhance the security is to use hardware devices. For example, the Barclays bank provides a hand-held card reader to the users. Before shopping in the net, users need to insert their credit card into the card reader, and input their (personal identification number) PIN code, then the card reader will produce a onetime security password, users can perform transactions only after the right password is input. Another method is to use the biometrics characteristic (e.g. voice, fingerprint, iris, etc.) for user authentication. For example, PayPal had tried to replace the single password verification by voice recognition to enhance the security of the Web site.

3.1.3 Block the phishing e-mails by various spam filters:

Phishers generally use e-mails as 'bait' to allure potential victims. SMTP (Simple Mail Transfer Protocol) is the protocol to deliver e-mails in the Internet. It is a very simple protocol which lacks necessary authentication mechanisms. Information related to sender, such as the name and email address of the sender, route of the message, etc., can be counterfeited in SMTP. Thus, the attackers can send out large amounts of spoofed e-mails which are seemed from legitimate organizations. The phishers hide their identities when sending the spoofed e-mails, therefore, if anti-spam systems can determine whether an e-mail is sent by the announced sender (Am I Whom I Say I Am?), the phishing attacks will be decreased dramatically.

3.1.4 Install online anti-phishing software in user's computers:

Despite all the above efforts, it is still possible for the users to visit the spoofed Web sites. As a last defense, users can install anti-phishing tools in their computers. The Anti-phishing tools in use today can be divided into two categories:

Category I:

When a user visits a Web site, the anti-phishing tool searches the address of that site in a blacklist stored in the database. If the visited site is on the list, the anti-phishing tool then warns the users. Tools in this category include Scam Blocker from the EarthLink Company, Phish Guard, and Net craft, etc. Though the developers of these tools all announced that they can update the blacklist in time, they cannot prevent the attacks from the newly emerged (unknown) phishing sites.

Category II:

This category of tools uses certain rules in their software, and checks the security of a Web site according to these rules. Examples of this type of tools include Spoof Guard developed by Stanford, Trust Watch of the Geo Trust, etc. Spoof Guard checks the domain name, URL (includes the port number) of Web site, it also checks whether the browser is directed to the current URL via the links in the contents of e-mails. If it finds that the domain name of the visited Web site is similar to a well-known domain name, or if they are not using the standard port, Spoof Guard will warn the users. In Trust Watch, the security of a Web site is determined by whether it has been reviewed by an independent trusted third party organization. Both Spoof Guard and Trust Watch provide a toolbar in the browsers to notify their users whether the Web site is verified and trusted.

3.1.5 Disadvantages of the Existing System

- It increases the overhead of the DNS systems and may cause problem for normal DNS queries, and furthermore, many phishing attacks simply do not require a DNS name.
- For phishing download detection, clever phishers may easily write tools which can mimic the behavior of human beings to defeat the detection.
- The Anti Phishing software tools cannot prevent the attacks from the newly emerged (unknown) phishing sites.
- It is easy to observe that all the above defense methods are useful and complementary to each other, but none of them are perfect at the current stage.

3.2 PROPOSED SYSTEM

3.2.1 Classification of the hyperlinks in the phishing e-mails:

In order to (illegally) collect useful information from potential victims, phishers generally tries to convince the users to click the hyperlink embedded in the phishing e-mail. A hyperlink has a structure as follows.

<a href="URI "> Anchor text <\a>

where ‘URI' (universal resource identifiers) provides the necessary information needed for the user to access the networked resource and 'Anchor text' is the text that will be displayed in user's Web browser.

Examples of URIs are:

- http://www.google.com
- https://www.icbc.com.cn/login.html
- ftp://61.112.1.90:2345.

'Anchor text' in general is used to display information related to the URI to help the user to better understand the resources provided by the hyperlink. In the following hyperlink, the URI links to the phishing archives provided by the APWG group, and its anchor text "Phishing Archive" informs the user what's the hyperlink is about. <a href http://www.antiphishing.org/phishing-archive.html">Phishing Archive </a>

Note that the content of the URI will not be displayed in user's Web browser. Phishers therefore can utilize this fact to play trick in their 'bait' e-mails. In the rest of the paper, we call the URI in the hyperlink the actual link and the anchor text the visual link. After analyzing the 203 (there are altogether 210 phishing e-mails, with 7 of them with incomplete information or with malware attachment and do not have hyperlinks).Phishing email archives from Sep. 21st 2003 to July 4th 2005 provided by APWG.

3.2.2 Advantages of Proposed System

- Link Guard works by analyzing the differences between actual links and visual links.
- It calculates the similarities of URI with a known trusted site.
- This algorithm can detect not only known but also unknown phishing attacks.
- It checks character wise so that it can be easily find phishing attack.
- It can detect up to 96% of unknown phishing attacks.
- Link guard is used for detecting the phishing attacks and also malicious links in web pages.

3.3 REQUIREMENTS SPECIFICATION

3.3.1 Hardware Requirements

The minimum hardware requirements are:

• Hard disk	:	20 GB and above
• RAM	:	256 MB and above
• Processor speed	:	1.6 GHz and above

3.3.2 Software Requirements

The minimum requirements for detection and prevention of phishing attacks are:

• Operating System	:	Windows 2000/XP or later
• Front End	:	Java(JDK version 8 or later)

• Database	:	MYSQL
• IDE	:	Eclipse

3.3.3 LANGUAGE SPECIFICATION

3.3.3.1 JAVA

Now-a-days all are familiar with Internet, the worldwide network of computers, which connects together thousands of computer all over the world. These network connections are increasing day by day in a rapid rate, so the network traffic is increasing at a pulse rate. Computers connected to the net are from many different manufacturers, running on different operating systems and they differ in architecture, computing power and capacity. By considering this point SUN Microsystems Corporation felt the need for a new programming language suitable for this heterogeneous Environment and java was the solution. This breaks barriers between different computers, chips and operating systems. Using java your application become compatible with all operating systems.

FEATURES OF JAVA:

- Simple
- Secure
- Portable
- Object oriented
- Robust
- Multithreaded

SIMPLE: It is simple for professional programmer to learn & they can use it effectively. If we already know object oriented programming, then learning java is very easy. It inherits syntax from c & object oriented features from C++, so if the user knows C\C++ then it will be a easy way to do effective java programming.

SECURE: As we know many people are effected by viral infection when they download an executable file or program. Rather than, virus programs we have malicious programs that can gather private information, such as credit card number, bank account balances & passwords by searching the contents of your computers local file system. Java has a better answer for this effect i.e., “FIREWALL” between networked application and your computer.

PORTABLE: As already we have discussed about compatibility of operating system, computers, chips. In Internet the programs have to be dynamically downloaded to all the various types of platforms. For this purpose java program will generate a byte code (which is not a executable code).Byte code is a highly optimized set of instructions designed to be executed by java run system, which is called as JVM (Java Virtual Machine).

OBJECT ORIENTED: Java is purely object oriented. The object model in java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

ROBUST: The ability to create robust programs was given a high priority in the design of java. To gain reliability, java restricts you in a few key areas, to force you to find your mistakes early in program development. At the same time, java frees you from having to worry about many of the most common causes of programming errors. Because java is a strictly typed language, it checks your code at Compile time. Java is robust for two reasons; they are Memory management & mishandled exceptional task in traditional programming environments.

3.3.3.2 JAVA DATABASE CONNECTIVITY:

JDBC is a Java API for executing SQL statements. (JDBC is often thought of as “Java Database Connectivity”) .It consists of a set of classes and interfaces written in the java programming language.

Using JDBC, it is easy to send SQL statements to virtually any relational database. In the other words, with the JDBC API, it is not necessary to write to one program to access a Sybase database, another program to access Informix database, another program to access Oracle database, and so on. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of JAVA and JDBC let’s a programmer writes it once and run it anywhere.

Java, being robust, secure, easy to use, easy to understand, and automatically downloadable on a network, is an excellent language basis for database applications.

JDBC extends what can be done in Java. For example, with Java and the JDBC API, it is possible to publish a web page, which contains an applet that uses information obtained from a remote database. With more and more Programmers using the Java Programming language, the need for easy database access from java is continuing to grow.

MIS managers like the combination of Java and JDBC because it makes disseminating information easy and economical, Businesses can continue to use their installed database and access information even if it is stored on different database management systems. Development time for new application is short. A programmer can write an application or an update once, put it on the server, and everybody has access to the latest version. A Low Level API and a Base for Higher-level APIs.

JDBC is a “Low level “ interface that means that it is used to invoke SQL commands directly, it works very well in this capacity and is easier to use than other database connectivity APIs, but it was designed to be a base upon which to build higher level interfaces and tools. A high level interface is “user-friendly”, using a more understandable or more convenient API that is translated behind the scenes into a low level interface such as JDBC. At the time of writing, two kinds of higher level APIs are under development of top of JDBC.

1) An embedded SQL for Java. At least one vendor plan to build this DBMS implement SQL, a language designed specifically for use with databases. JDBC requires that the SQL statements be passed as Strings to Java methods. The embedded SQL preprocessor then translates this JAVA/SQL mix into Java with JDBC calls.

2) A direct mapping of relational database tables to Java classes. In this “object/relational” mapping, each row of the table becomes an instance of that class, and each column value corresponds to an attribute of that instance. Programmers can then operate.

ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.

A Java API like JDBC is needed in order to enable a “Pure Java” solution. When ODBC is used, the ODBC Driver Manager and drivers must be manually installed on every client machine. When the JDBC driver is written completely in Java, however, JDBC code is automatically installable, portable, and secure on all platforms from network computers to mainframes.

3.3.3.4 Database Management System (DBMS):

A Database is an integrated collection of user related data stored with minimum redundancy, serves many users/applications quickly and efficiently.A database system is basically a computerized record keeping system, i.e. it is a computerized system whose overall purpose is to maintain information and make that information available on demand.

DBMS is a collection of inter-related data and set of programs that allow several users to access and manipulate data. Its main purpose is to provide users with an abstract view of the data, i.e. the system hides certain details of how the data is stored and maintained.

Database Management System is divided into 4 main components

- Database
- Hardware
- Software
- User

Database: It consists of collection of persistent data that is used by the application system.

Hardware: The processor(s) and associated main memory that are used to support the execution of database system software..

Software: The layer between the physical database and the users that handles all requests from the user for access to the database.

User: There are three types of users

- Application Programmers
- End User

- Database Administrator (DBA)

3.3.3.4.1 TYPES OF DBMS

There are four major categories of DBMS data models.

- ➢ Hierarchical
- ➢ Network
- ➢ Inverted
- ➢ Relational

3.3.3.4.1.1 RELATIONAL DATABASE MANAGEMENT SYSTEMS

Database Management System has evolved from hierarchical to network to relational models. Today, the most widely accepted database model is the relational model. The relational database management system uses only its relational capabilities to manage the information stored in the database. The relational model has three different aspects.

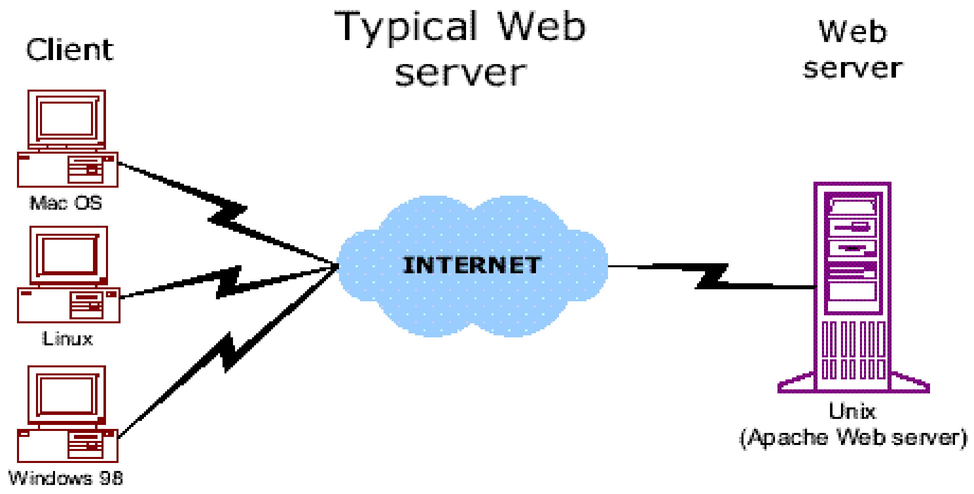
- Structures
- Operation
- Integrity rules

Structures: They are well-defined objects that store the data of a database structure and the data contained within them can be manipulated by operations.

Operations: They are clearly defined actions that allow users to manipulate the data and structures of a database. The operations on a database must adhere to a predefined set of integrity rules.

3.3.3.5 JAVA SERVER PAGE (JSP):

Java Server Pages (JSP) is a new technology for web application development that has received a great deal of attention since it was first announced.



A JSP is similar in design and functionality to java servlet. It is called by the client to provide a web service, the nature of which depends on the J2EE application. However, a JSP differs from a servlet in the way in which the JSP is written. Java Servlet is written using Java programming language and responses are encoded as an output string object that is passed to the println() method. In contrast a JSP is written in HTML, XML, or in the client's format that is interspersed with scripting elements, directives, and actions comprised of Java Programming language and JSP syntax.

There are three methods that are automatically called when the JSP is requested and the JSP terminates normally. These are the jspInt() method, the jspDestroy() method, and the service() method.

The jspInt () method is called first when the jsp is requested and is used to initialize objects and variables that are used throughout the life of the JSP. The jspDestroy() method is automatically called when the JSP terminates normally. It isn't called when the JSP abruptly terminates. The service () method is automatically called and retrieves connection to HTTP.

The jsp programs are executed by a JSP virtual machine that runs on a web server. Therefore you will need to have access to a JSP virtual machine to run your JSP program. One of the most popular JSP virtual machines is TOMCAT.

1. JSP is Java-based, and Java is well-suited for enterprise computing.

In fact, JSP is a key part of the Java 2 Enterprise Edition (J2EE) platform and can take advantage of the many Java Enterprise libraries, such as JDBC, JNDI, and Enterprise Java Beans.

2. JSP supports a powerful model for developing web applications that separates presentation from processing.

In the early days of the Web, the only tool for developing dynamic web content was the Common Gateway Interface (CGI). CGI outlined how a web server made user input available to a program, as well as how the program provided the web server with dynamically generated content to send back. CGI scripts were typically written in Perl. (In fact, CGI Perl scripts still drive numerous dynamic web sites.) However, CGI is not an efficient solution. For every request, the web server has to create a new operating-system process, load a Perl interpreter and the Perl script, execute the script, and then dispose of the entire process when it's done. To provide a more efficient solution, various alternatives to CGI have been added to programmers' toolboxes over the last few years: FastCGI, for example, runs each CGI program in an external permanent process (or a pool of processes). In addition, mod_perl for Apache, NSAPI for Netscape, and ISAPI for Microsoft's IIS all run server-side programs in the same process as the web server itself. While these solutions offer better performance and scalability, each one is supported by only a subset of the popular web servers.

The Java Servlet API, introduced in early 1997, provides a solution to the portability issue. However, all these technologies suffer from a common problem: HTML code embedded inside programs. If you've ever looked at the code for a servlet, you've probably seen endless calls to out.println( ) that contain scores of HTML tags.

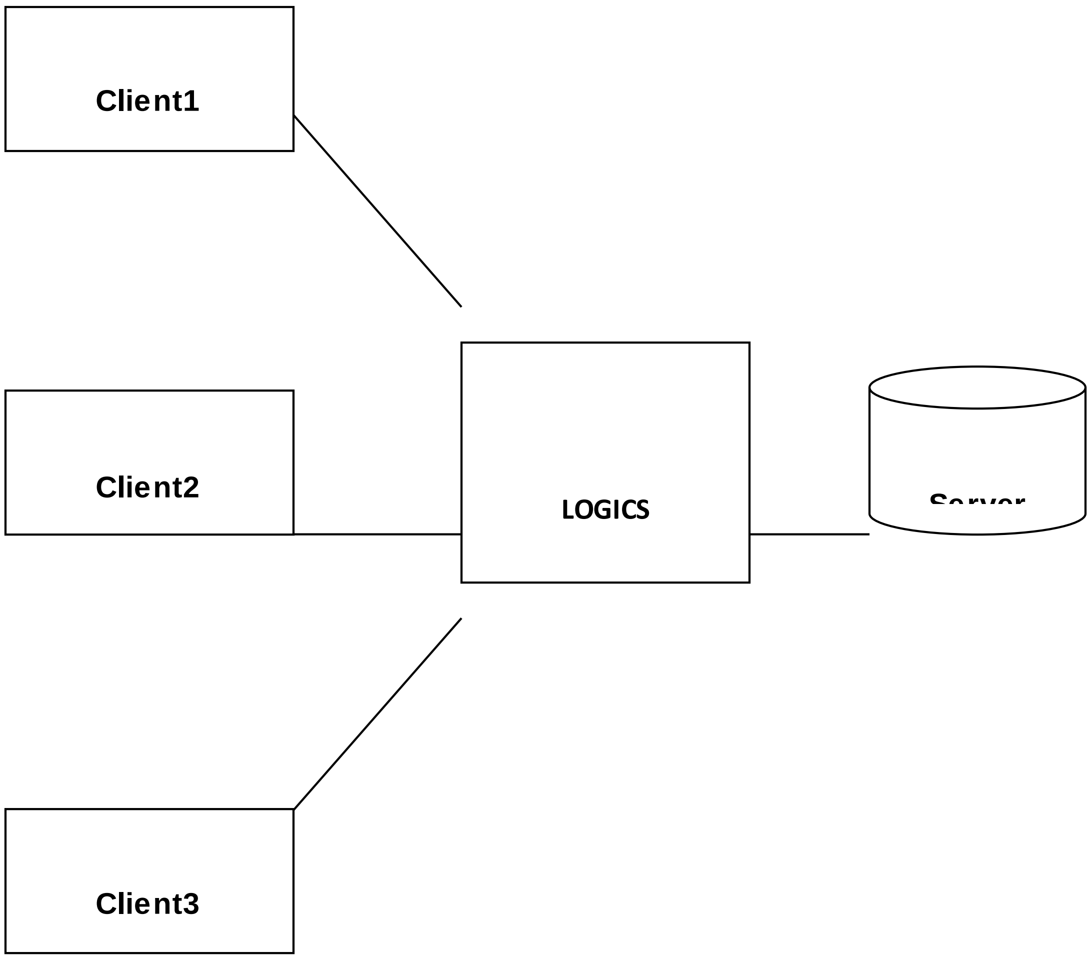
For the individual developer working on a simple web site this approach may work fine, but it makes it very difficult for people with different skills to work together to develop a web application. This is becoming a significant problem.

As web sites become increasingly complex and are more and more critical to the success of an organization, the appearance and usability of the web interface becomes paramount. New client technologies, such as client-side scripts and DHTML, can develop more responsive and interactive user interfaces, style sheets can make it easier to globally change fonts and colors, and images can make the interface more appealing. At the same time, server-side code is getting more complex, and demands for reliability, performance, and fault tolerance are increasing. The growing complexity of web applications requires a development model that allows people with different skills to cooperate efficiently.

Java Server Pages provides just such a development model, allowing web page authors with skills in graphics, layout, and usability to work in tandem with programmers who are experienced in server-side technologies such as multithreading, resource pooling, databases, and caching. While there are other technologies, such as ASP, PHP and ColdFusion, that support similar development models, none of them offers all the advantages of JSP.

3.3.6 JSP ARCHITECTURE:

In the process of dynamic content generation, Sun Microsystems has introduced with a new concept as "Logic separation" from client.



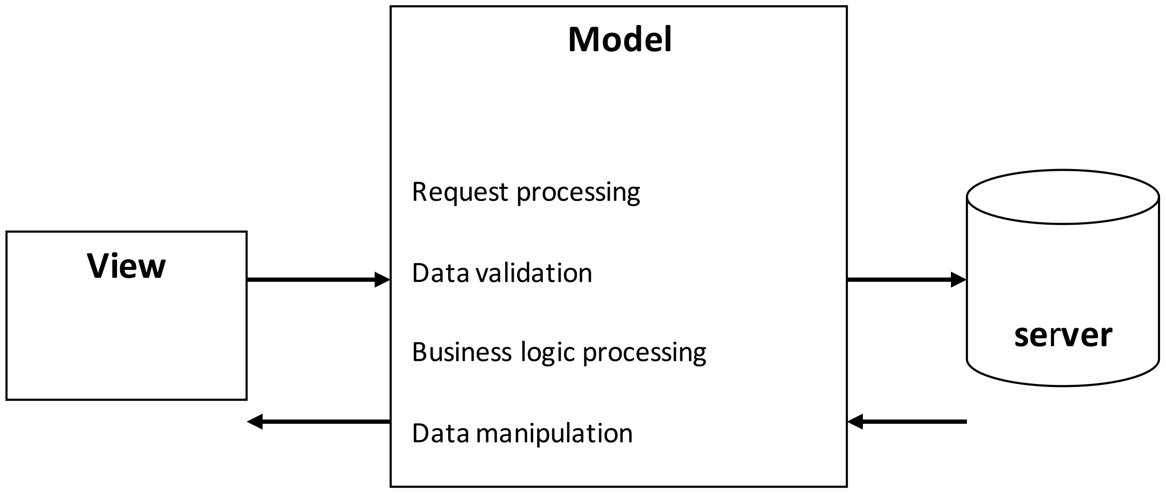
Logic which is required is written at one place and can be accessed by all the clients which are at different places when they require. So, in order to implement this, we have two Architectures known as

▪ Model 1
▪ Model 2

Model-1 Architecture:

In Model-1 Architecture, presentation logics and Bussiness logics are combined. When these two are combined each presentation logic requires copy of business logic so that the model and server should work towards this, such thing is time and resource consume, so in order to overcome this Model 2 Architecture has been introduced.

Model-1 Architecture is called as Java Bean.



Model-2 Architecture:

Model-2 Architecture is also called as Java Strut. In this Model, presentation and business logics are separated. We have three parts in this,

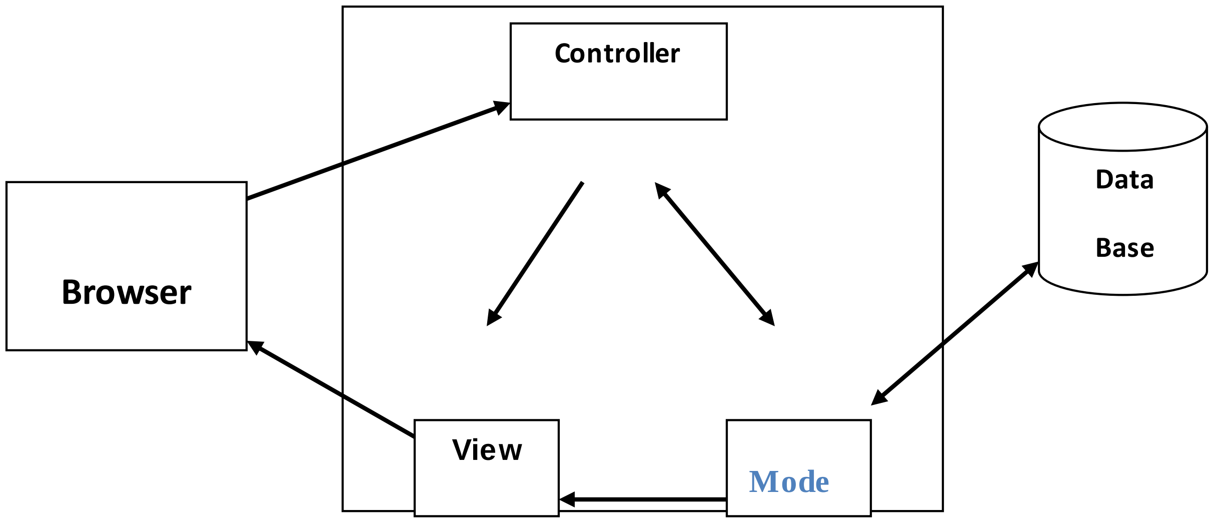
- View
- Controller

- Model

Model: contains business logics which represent application state along with database

View: JSP acts a view which receives data from the model this view doesn’t contain any data manipulation logics.

Controller: takes request from client, process the Model and provide information to different JSP page and passing control to different views.



3.3.7 JAVA SERVLETS

A Java Servlet is a server-side program that is called by the user interface or another J2EE component and contains the business logic to process a request. In this the implicit and the explicit data is sent from a client to a server-side program in the form of a request that is processed and another set of explicit implicit data is returned.

Explicit data is information received from the client that is typically either entered by the user in to user interface or generated by the user interface itself. Implicit data is HTTP information that is generated by the client rather than the user.

Advantages of java servlets

- Only one copy of a java servlet is loaded in to the JVM no matter the number of simultaneous requests.
- A java servlet has persistence. This means that the servlet remains alive after the request.

3.3.8 APACHE TOMCAT SERVER

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the Java Server Pages specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Environment

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets. The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments. Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

Product features

- Tomcat 3.x (initial release)
- implements the Servlet 2.2 and JSP 1.1 specifications
- servlet reloading
- basic HTTP functionality

Tomcat 4.x

- implements the Servlet 2.3 and JSP 1.2 specifications
- servlet container redesigned as Catalina
- JSP engine redesigned as Jasper
- Coyote connector
- Java Management Extensions (JMX), JSP and Struts-based administration

Tomcat 5.x

- implements the Servlet 2.4 and JSP 2.0 specifications
- reduced garbage collection, improved performance and scalability
- native Windows and Unix wrappers for platform integration

- faster JSP parsing

3.3.9 JAVA MAIL API

Email is probably the most widely used methods of communication. A J2EE application is able to send and receive email messages through the use of the Java mail API. It is protocol independent and can send and receive messages created by a J2EE application via email using existing email protocols.

It provides a set of abstract classes defining objects that comprise a mail system. The API defines classes like message, store and transport. The API can be extended and can be sub classed to provide new protocols and to add functionality when necessary.

JAVA MAIL LAYERED ARCHITECTURE

The Java Mail architectural components are layered as shown below:

- The abstract layer declares classes, interfaces and abstract methods intended to support mail handling functions that all mail system support
- The internet implementation layer part of the abstract layer using Internet standards

Java mail uses the Java Beans Activation Frame work (JAF) in order to encapsulate message data and to handle commands intended to interact with the data.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

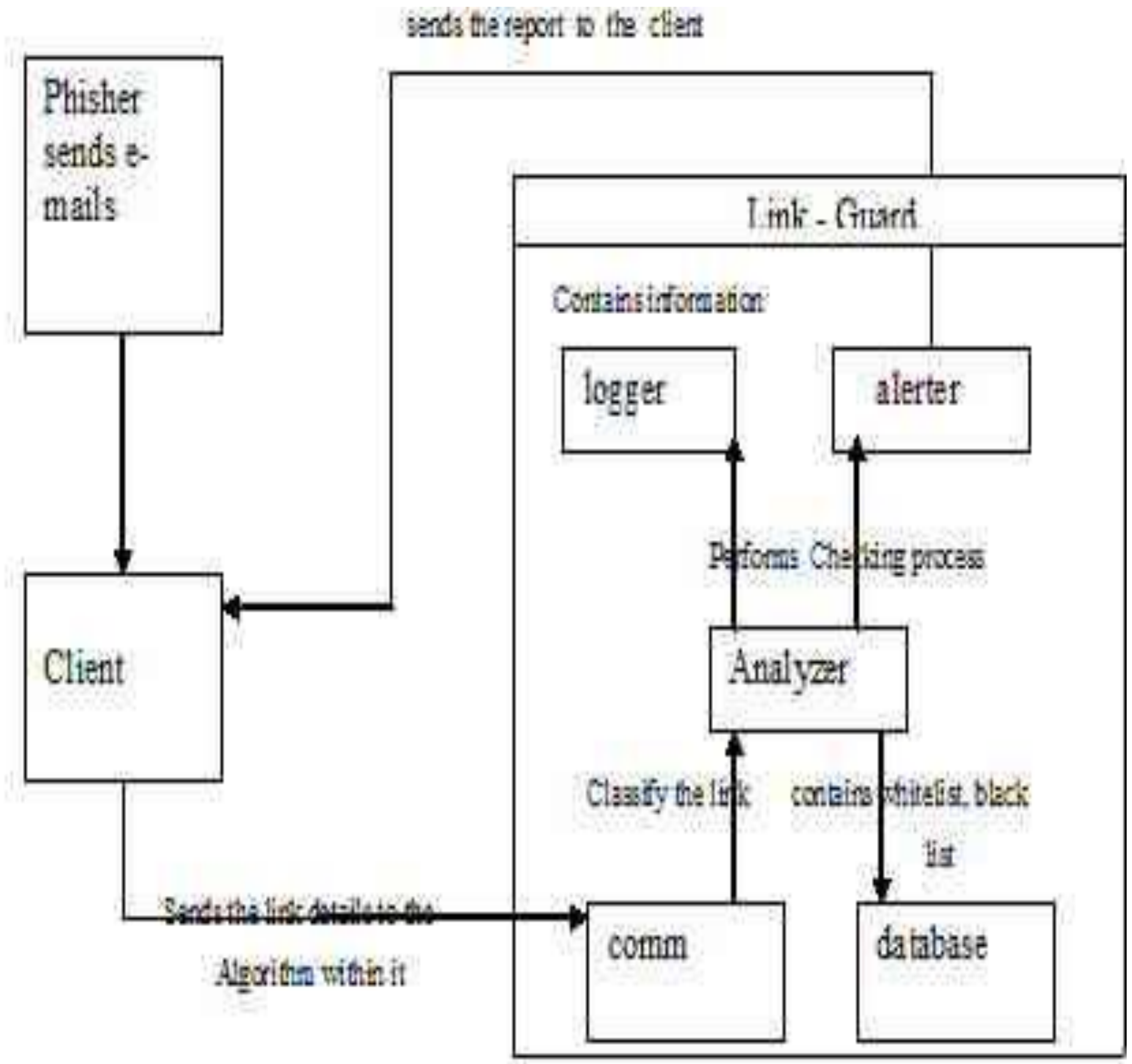

sends the report to the client
Phisher sends e-mails
Link - Guard
Contains information
logger
alerter
Performs Checking process
Client
Analyzer
Classify the link
contains whitelist, black list
Sends the link details to the Algorithm within it
comm
database

Figure 4.1 Architecture of Proposed System

The above figure contains the implementation of the Link Guard algorithm. It is possible for the user to add domain names and categorize them as either white list or black list under settings. Whenever a mail is detected as phishing the domain name in that mail automatically gets added as black list. The Link Guard algorithm checks if the domain names fall under any of the 5 categories of hyperlinks for phishing emails. It also refers to the database of black and white list entries and sets the status of the mail as either **Phishing** or **Non-Phishing.** Once the mail is categorized as Phishing the user can take care that he does not open the link or submit any personal, critical information on to the website.

4.2 UNIFIED MODELING LANGUAGE

UML stands for Unified Modeling Language are a third generation method for specifying, visualizing and documenting the artifacts of an object oriented system under development. Object modeling is the process by which the logical objects in the real world (problem space) are represented (mapped) by the actual objects in the program (logical or a mini world). This visual representation of the objects, their relationships and their structures is for the ease of understanding. This is a step while developing any product after analysis.

The goal from this is to produce a model of the entities involved in the project which later need to be built. The representations of the entities that are to be used in the product being developed need to be designed.

Software design is a process that gradually changes as various new, better and more complete methods with a broader understanding of the whole problem in general come into existence.

4.2.1 USECASE DIAGRAM

Use case diagram consists of use cases and actors and shows the interaction between them. The key points are:

- The main purpose is to show the interaction between the use cases and the actor.
- To represent the system requirement from user’s perspective.
- The use cases are the functions that are to be performed in the module.

An actor could be the end-user of the system or an external system.

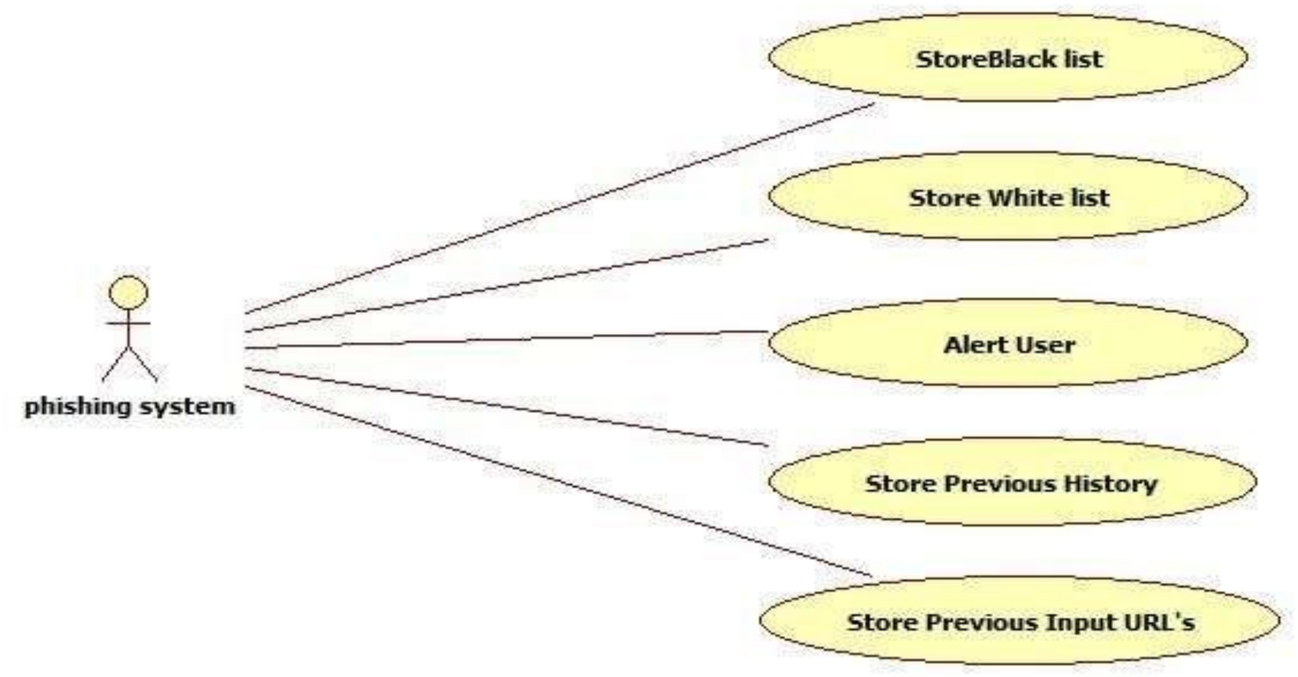


Figure 4.2 Use case Diagram for phishing system

4.3.2 CLASS DIAGRAM

Class Diagram consists of the classes and the objects and the interaction between them. It mainly deals with the interaction between classes in the system, their behavior and properties of the system. Apart from classes this also provides inheritance relationships in the project. Class diagrams consist of basically two parts: first one is the member variables and class variables and the second part consists of the total number of methods available in the class.

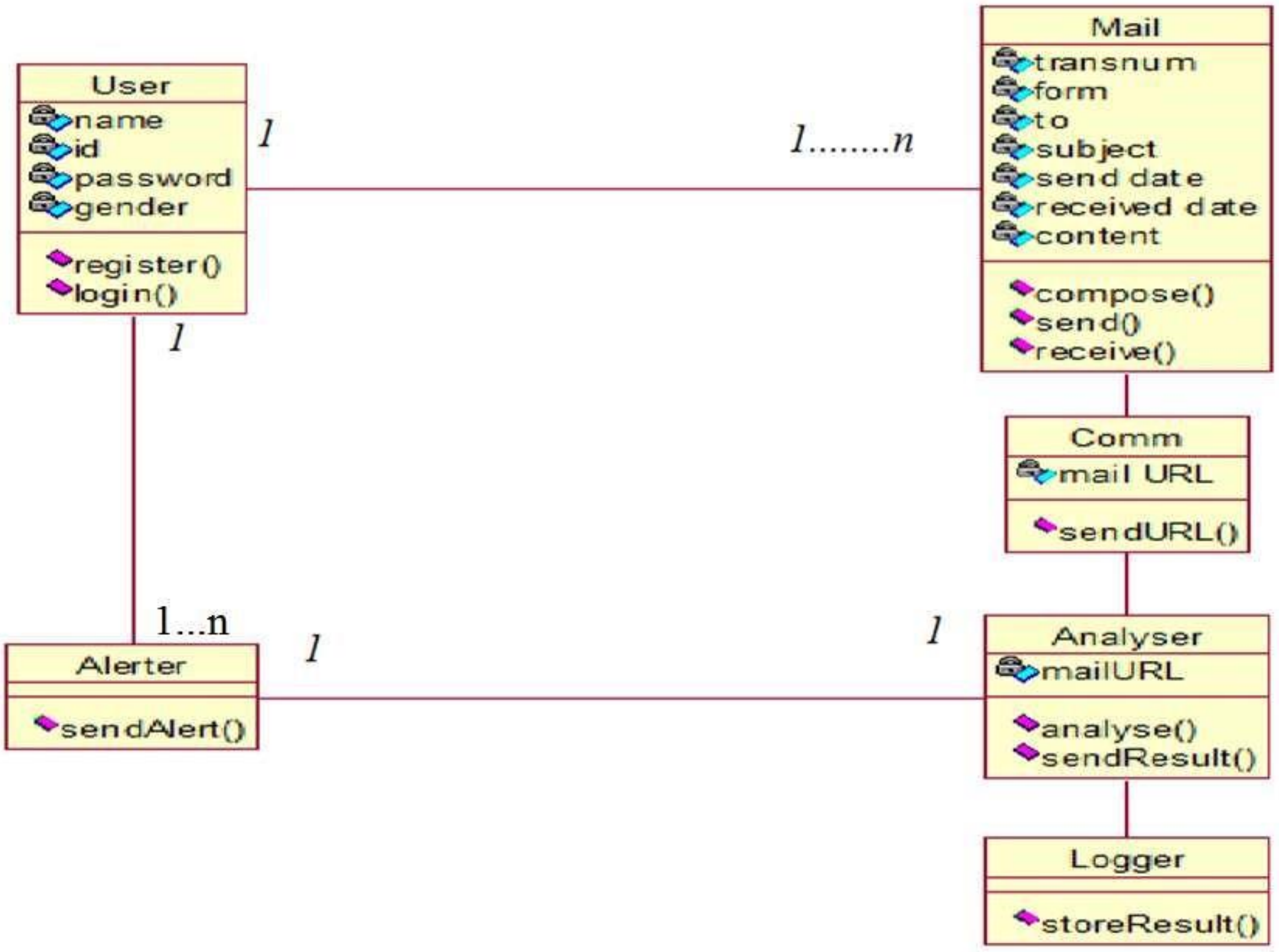


Figure 4.3 Class Diagram

4.3.3 SEQUENCE DIAGRAM

The purpose of sequence diagram is to show the flow of functionality through a use case. In other words, we call it a mapping process in terms of data transfers from the actor through the corresponding objects.

The key points are:

- The main purpose is to represent the logical flow of data with respect to a process
- A sequence diagram displays the objects and not the classes.

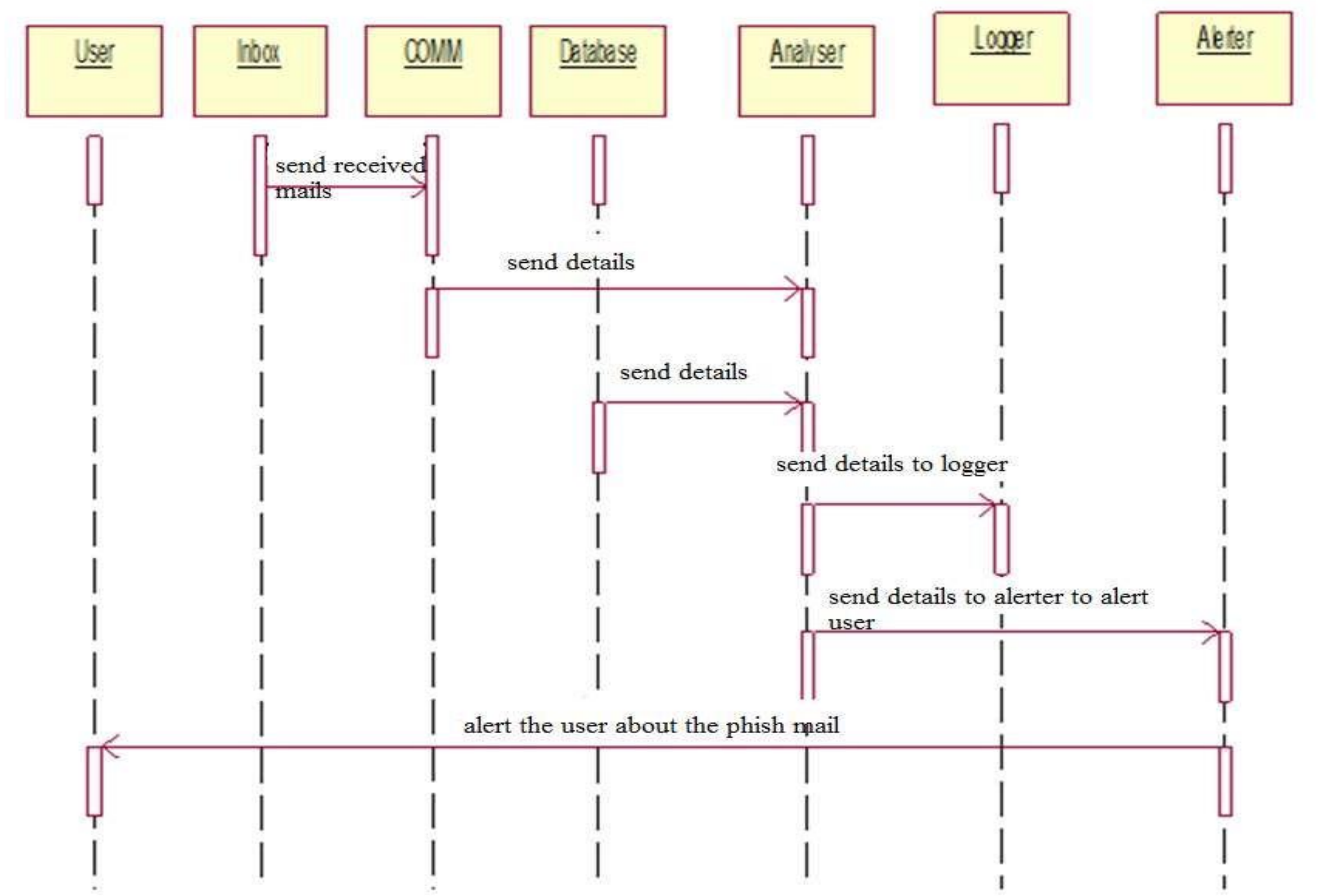


Figure 4.4 Sequence Diagram

CHAPTER 5

MODULE DESCRIPTION

5.1 MODULES

There are three modules involved in this project:

- Creation of a mail system and database operations
- Composes, send and receive a mail
- Implementation of the Link Guard algorithm

5.1.1 CREATION OF A MAIL SYSTEM AND DATABASE OPERATIONS

This module deals with the user interface for the home page, sign-in, sign-up and forgot your password pages. This module enables a new user to Sing-Up. It also enables an existing user to Sign-In. The user may use the Forget password link if he did forget his password. The password is retrieved on the basis of security question and answer given by the user. Database operation manages the users. Every time a new user signs in his details are written in to the database.

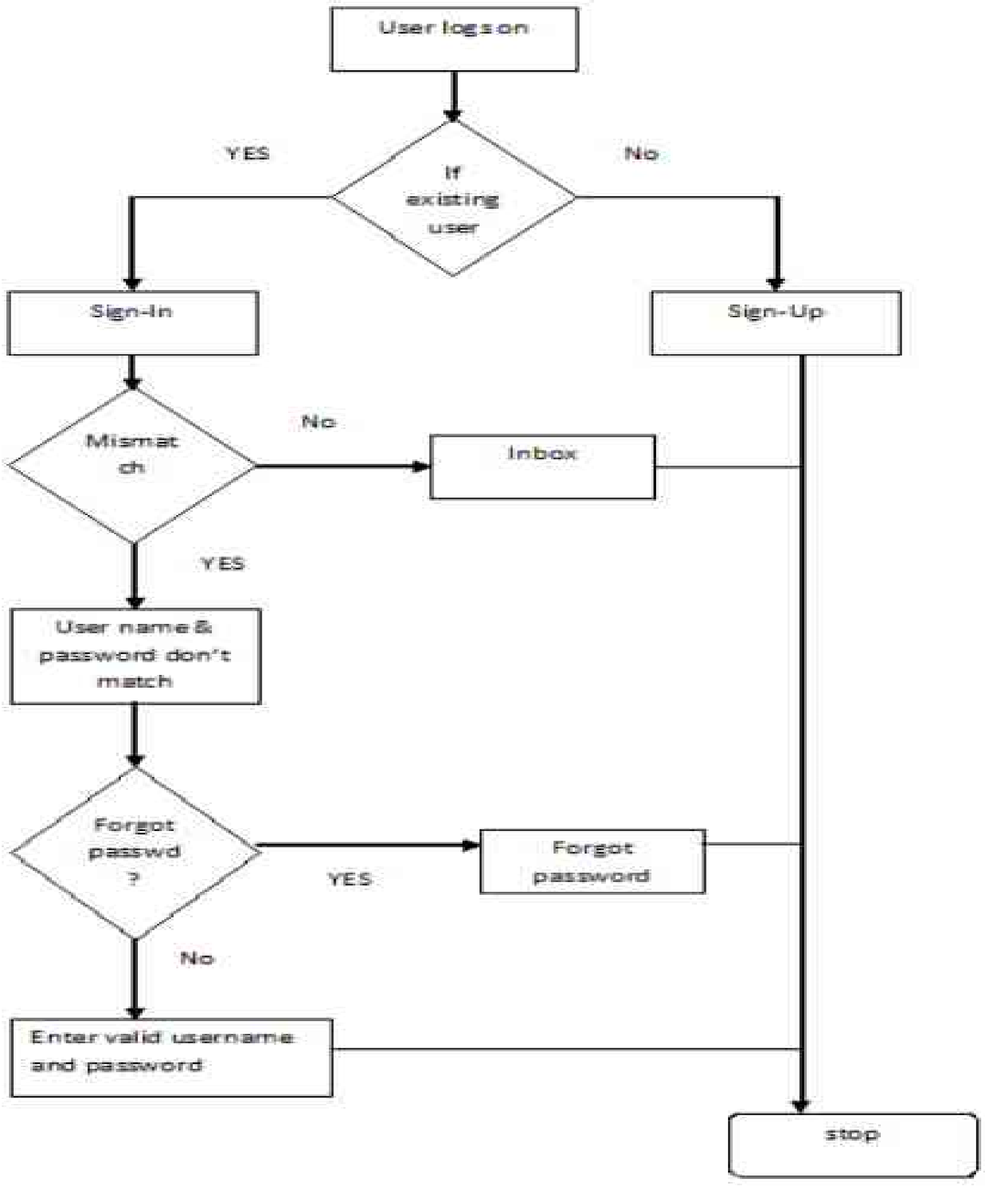


Figure 5.1.1 Creation of mail system and database operations

5.1.2 COMPOSES, SEND AND RECEIVE A MAIL

This module enables the user to compose and send a mail. It also allows the user to read a received mail. Once a mail is sent the date and the subject of the mail gets displayed. The received mail can be checked if it is phishing or not, the implementation of which is given in the next module. The compose mail option contains an option for spoof id. The spoof id allows the mail of the composer to be delivered with a different from address. This is being incorporated to demonstrate the Link Guard algorithm.

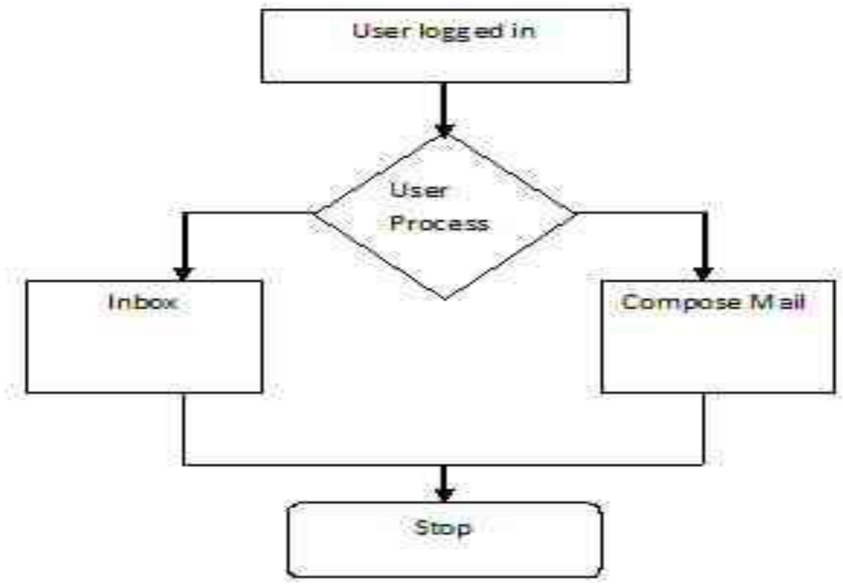


Figure 5.1.2 Composes, Send and Receive a Mail

5.1.3 IMPLEMENTATION OF THE LINK GUARD ALGORITHM

The module contains the implementation of the Link Guard algorithm. It is possible for the user to add domain names and categorize them as either white list or black list under settings. Whenever a mail is detected as phishing the domain name in that mail automatically gets added as black list. The Link Guard algorithm checks if the domain names fall under any of the 5 categories of hyperlinks for phishing emails. It also refers to the database of black and white list entries and sets the status of the mail as either **Phishing** or **Non-Phishing.** Once the mail is categorized as Phishing the user can take care that he does not open the link or submit any personal, critical information on to the website.

Comm.: This collects the information of the input process, and sends these related information's to the Analyzer.

Database: Store the white list, blacklist, and the user input URLs.

Analyzer: It is the key component of Link Guard, which implements the Link guard algorithm; it uses data provided by Comm. and Database, and sends the results to the Alert and Logger modules.

Alerter: When receiving a warning message from Analyzer, it shows the related information to alert the users and send back the reactions of the user back to the Analyzer.

Logger: Archive the history information, such as user events, alert information, for future use.

Link Guard works by analyzing the differences between the visual link and the actual link. It also calculates the similarities of a URI with a known trusted site. The algorithm is illustrated in Fig.1.The following terminologies are used in the algorithm.

v_link: visual link;

a_link: actual_link;

v_dns: visual DNS name;

```
a_dns: actual DNS name;
sender_dns: sender's DNS name.
1 .int Link Guard (v_link, a_link} {
2. v_dns = GetDNSName(v_link);
3. a_dns = GetDNSName(a_link);
4. if ((v_dns and a_dns are not
5. empty) and (v_dns != a_dns))
6. Return PHISHING;
 7. if (a_dns is dotted decimal)
 8. return POSSIBLE_PHISHING;
 9. if (a_link or v_link is encoded)
10.{
11. v_link2 = decode (v_link);
 12. A_link2 = decode (a_link);
 13. return Link Guard(v_link2, a_link2);
14. }
15. /*analyze the domain name for possible phishing*/
16.  if (v_dns is NULL)
17. return AnalyzeDNS(a_link);
18. }
```

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

```
 19. int AnalyzeDNS (actual link) {
/return PHISHING;
*Analyze the actual DNS name according
to the blacklist and whitelist
*/
20. if (actual_dns in blacklist)
21. if (actual_dns in whitelist)
22. Return NOTPHISHING;
 23. return Pattern Matching(actual_link);
}
24. int Pattern Matching(actual_link){
25. if (sender_dns and actual_dns are different)
 26. return POSSIBLE_PHISHING;
27. for (each item prev_dns in seed_set)
28. {
 29. bv = Similarity(prev_dns, actual_link);
30. if (bv == true)
31. return POSSIBLE_PHISHING;
32. }
33. return NO_PHISHING;
}
34. float Similarity (str, actual_link) {
35. if (str is part of actual_link)
36. Return true;
37 int maxlen = the maximum string
38. Lengths of str and actual_dns;
39 int minchange = the minimum number of
40. Changes needed to transform str
41. To actual_dns (or vice verse);
42.  if (thresh<(maxlen-minchange)/maxlen<1)
43. return true/
44. return false;
45}
```

The subroutines used in the Link Guard algorithm. (Categories 3 and 4), we first decode the links, then recursively call Link Guard to return a result (lines 8-13). When there is no destination information (DNS name or dotted IP address) in the visual link (category 5), Link Guard calls *Analyze DNS* to analyze the actual DNS (lines 16 and 17). Link Guard therefore handles all the 5 categories of phishing attacks. *Analyze DNS* and the related subroutines are depicted in Fig.2. In *Analyze DNS*, if the actual DNS name is contained in the blacklist, then we are sure that it is a phishing attack (lines18 and 19). Similarly, if the actual DNS is contained in the White list, it is therefore not a phishing attack (lines 20 and21). If the actual DNS is not contained in either white list or blacklist, Pattern Matching is then invoked (line 22).

Pattern Matching is designed to handle unknown attacks (blacklist/whitelist is useless in this case). For category 5 of the phishing attacks, all the information we have is the actual link. From the hyperlink (since the visual link does not contain DNS or IP address of the destination site), which provide very little information for further analysis. In order to resolve this problem, we try two methods: First, we extract the sender e-mail address from the e-mail. Since phishers generally try to fool users by using (spoofed) legal DNS names in the sender e-mail address, we expect that the DNS name in the sender address will be different from that in the actual link. Second, we proactively collect DNS names that are manually input by the user when she surfs the Internet and store the names into a *seed set*, and since these names are input by the user by hand, we assume that these names are trustworthy. Pattern Matching then checks if the actual DNS name of a hyperlink is different from the DNS name in the sender's address (lines 23 and 24), and if it is quite similar (but not identical) with one or more names in the *seed set* by invoking the Similarity (lines 25-30) Procedure.

Similarity checks the maximum likelihood of actual DNS and the DNS names in *seed set*. As depicted in Fig. 2, the similarity index between two strings are determined by calculating the minimal number of changes (including insertion, deletion, or revision of a character in the string) needed to transform a string to the other string. If the number of changes is 0, then the two strings are identical; if the number of changes is small, then they are of high similarity; otherwise, they are of low similarity. For example, the similarity index of 'microsoft' and 'micr0s0ft' is 7/9 (since we need change the 2 '0's in micr0s0ft to 'o'. Similarly, the similarity index of 'paypal' and 'paypal-cgi' is 6/10 (since we need to remove the last 4 chars from paypal-cgi), and the similarity index of '95559' and '955559' is 5/6 (since we need to insert a '5' to change '95559' to '955559'). If the two DNS names are similar but not identical, then it is a possible phishing attack. For instance, Pattern Matching can easily detect the difference between www.icbc.com.cn (which is a good e-commence Web site) and www.1cbc.com.cn (which is a phishing site), which has similarity index 75%.

Note that Pattern Matching may treat www.1cbc.com.cn as a normal site if the user had never visit www.1cbc.com.cn before. This false negative, however, is unlikely to cause any severe privacy or financial lose to the user, since she actually does not have anything to lose regarding the Web site www.icbc.com.cn (since she never visits that Web site before)!

False positives and false negatives handling:

Since Link Guard is a rule-based heuristic algorithm, it may cause false positives (i.e., treat non-phishing site as phishing site) and false negatives (i.e., treat phishing site as non-phishing site). In what follows, we show that Link Guard may result in false positives but is very unlikely to cause harmful false negatives. For phishing attacks of category 1, we are sure that there is no false positives or false negatives, since the DNS names of the visual and actual links are not the same. It is also easy to observe that Link Guard handles categories 3 and 4 correctly since the encoded links are first decoded before further analysis.

For category 2, Link Guard may result in false positives, since using dotted decimal IP addresses instead of domain names may be desirable in some special circumstances (e.g. when the DNS names are still not registered). For category 5, Link Guard may also result in false positives. For example, we know that both 'www.iee.org' and 'www.ieee.org' are legal Web sites. But these two DNS names have a similarity index of 3/4, hence is very likely to trigger a false positive. When it is a possible false positive, Link Guard will return a POSSIBLE PHISHING. In our implementation (which will be described in the next section), we leverage the user to judge if it is a phishing attack by prompting a dialogue box with Detailed information of the hyperlink. The rationale behind this choice is that users generally may have more knowledge of a link than a computer in certain circumstances (e.g., the user may know that the dotted decimal IP address is the address of his friend's computer and that www.iee.org is a respected site for electrical engineers).

For category 5, Link Guard may also result in false negatives. False negatives are more harmful than false positives, since attackers in this case will succeed in leading the victim to the phishing sites. For instance, when the sender's e-mail address and the DNS name in the actual link are the same and the DNS name in the actual link has a very low similarity index with the target site, Link Guard will return

NO PHISHING. For instance, Pattern Matching will treat the below link as NO PHISHING.
<a href="http://fdic-secure.com/
application.htm"> Click here </a>
with securehq@fdic-secure.com" as the sender address. We note that this kind of false negatives is very unlikely to result in information leakage, since the end user is very
unlikely to have information the attack interested (since the DNS name in this link is not similar with any legal Web sites).

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

At present generation attackers are more in networks, phishing has become major security problem, causing many losses by hacking he legal data that are used by the user. Phishers set their own fake websites which exactly looks like the original website including applying DNS server name, setting up web server and creating web pages similar to destination website. So in this paper we had designed link guard algorithm which is a character based. So in this paper we had designed link guard algorithm which is a character based. It had detected many attackers by using APWG (anti phishing working group). Link guard is implemented for all OS, it can detect up to 96% of unknown phishing attacks. Link guard is used for detecting the phishing attacks and also malicious links in web pages.

6.2 FUTURE ENHANCEMENT

As future work, we are trying to detect Phishing attack with more advanced techniques like Machine Learning (ML) and Artificial Intelligence (AI).Further improvement can be done to other sources of phishing attacks other than email. We are planning to innovate advanced algorithm which will probably having more efficient way than Link guard algorithm to detect and prevent phishing attack.

APPENDIX 1

SAMPLE CODING

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.*;
import javax.servlet.http.*;



public class Domain extends HttpServlet
{
        Exception e1=null;
public void doPost(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{
PrintWriter out = response.getWriter();
String name=request.getParameter("t1");
String type=request.getParameter("t2");
boolean result=check(name);
if(result)
{
            RequestDispatcher rq=request.getRequestDispatcher("settings.jsp");
           request.setAttribute("msg","The Domain name already exist");
           rq.forward(request,response);
}
else
{
           if(save(name,type))
           {
           RequestDispatcher rq=request.getRequestDispatcher("settings.jsp");
           request.setAttribute("msg","The Domain name successfully saved");
           rq.forward(request,response);
           }
else
{
                      RequestDispatcher rq=request.getRequestDispatcher("settings.jsp");
                      request.setAttribute("msg",e1);
                      rq.forward(request,response);
           }
}
}
```

```
public boolean check(String name)
{
        boolean exist=false;
        try
        {
                DriverManager.registerDriver( new oracle.jdbc.driver.OracleDriver()
);
                Connection conn = DriverManager.getConnection(
"jdbc:oracle:thin:@"+getServletConfig().getServletContext().getInitParameter("ser
ver"),"system","redhat" );
                PreparedStatement st = conn.prepareStatement("Select * from LIST
WHERE name= '"+name+"'");
                ResultSet rs = st.executeQuery();
                while(rs.next())
                {
                        exist=true;
                }


                st.close();
                  conn.close();
}
catch(Exception e)
{
        e.printStackTrace();
        return false;
}


return exist;
}
public boolean save(String name,String type)
{
        try
        {
                DriverManager.registerDriver( new oracle.jdbc.driver.OracleDriver()
);
                Connection conn = DriverManager.getConnection
("jdbc:oracle:thin:@"+getServletConfig().getServletContext().getInitParameter("se
rver"),"system","redhat" );
                PreparedStatement st = conn.prepareStatement("Select * from LIST");
                String query="insert into LIST values('"+name+"','"+type+"')";
                st.executeUpdate(query);
                st.close();
                conn.close();
        return true;
        }
        catch(Exception e)
        {
                e1=e;
```

```
                    return false;
               }
     }
          }
```

REFERENCES