Detection of phishing websites using ML

ABSTRACT

Web Phishing appeals to the user to connect with the fake site. The main goal of this attack is to rob the user of sensitive information. The intruder builds websites similar to those that look like the original website. It allows attackers to access confidential information such as username, password, details of credit cards etc. This paper aims to review many of the phishing detection strategies recently suggested for the website. This will also provide a high-level description of various forms of phishing detection techniques. Here proposed a multidimensional element phishing recognition approach dependent on a quick discovery method by using deep learning (MFPD). In the initial step, character succession highlights of the given URL are separated and utilized for snappy characterization by profound learning, and this progression doesn't need outsider help or any earlier information about phishing. In the subsequent advance, we consolidate URL measurable highlights, website page code highlights, site page content highlights and the brisk characterization consequence of profound learning into multidimensional highlights. The methodology can diminish the identification time for setting an edge. Testing on a dataset containing a huge number of phishing URLs and genuine URLs, the exactness arrives at 98.99%, and the bogus positive rate is just 0.59%. By sensibly changing the limit, the test results show that the discovery effectiveness can be improved

List of figures/diagrams/graphs

TABLE OF CONTENTS

1. INTRODUCTION

1.1 About the project

The phishing website detection based on machine learning is a hotspot of current phishing website detection research. The results of machine learning methods usually depend on the quality of the extracted features. The focus of current research is on how to extract and select more effective features before processing them.

1.2 Objective

- Phishing website recognition framework gives solid security system to distinguish and forestall phishing areas from arriving at client. This venture presents a basic and compact way to deal with identify parodied pages and fathom security vulnerabilities utilizing Machine Learning. It very well may be effectively worked by anybody since all the significant errands are going on in the backend.

2. SYSTEM ANALYSIS

2.1 Existing System

Huaping Yuan, Xu Chen and Yukun Li et al. [2] uses different algorithms for detecting the phishing websites. Various ML algorithms on phishing detection including kNearest Neighbor(KNN), Logistic Regression(LR), Random Forest(RF), Decision Tree(DT), Gradient Boosting Decision Tree(GBDT), XGBoost(XGBST), and Deep Forest(DF).The authors introduce the statistical features and lexical features of URLs and links

2.2 Proposed System

A multidimensional component phishing identification approach dependent on a quick recognition technique by utilizing profound learning. In the _rst step, character grouping highlights of the given URL are removed and utilized for snappy classi_cation by profound learning, and this progression doesn't require thirdparty help or any earlier information about phishing. In the subsequent advance, we consolidate URL factual highlights, page code highlights, website page content highlights, and the brisk classi_cation aftereffect of profound learning into multidimensional highlights. The methodology can decrease the location time for setting an edge. Testing on a dataset containing a large number of phishing URLs and genuine URLs, the precision arrives at 98.99%, and the bogus positive rate is just 0.59%. By sensibly altering the limit, the exploratory outcomes show that the recognition ef_ciency can be improved.

.

2.2.1 Details

2.2.2 Impact on Environment

impact on environment (not OS or SW used), Examples – Reduction in global warming, reduce pollution, simplicity of usage, time reduction etc.,

2.2.3 Safety

Impact on various areas mentioned (but not limited to) Security (data, network, information), privacy etc.,

2.2.4 Ethics

General SW ethics for building an application or solution like (but not limited to) – does not harm any person (physically or virtually), securing privacy information of the resources using application (secure login, not exposing personal details in any form) etc.,

2.2.5 Cost

Cost of development, usage, maintenance etc.,

Cost reduction due to implementation of the project in production

2.2.6 Type

Standalone

2.2.7 Standards

STRUCTURE OF PROJECT (SYSTEM ANALYSIS)

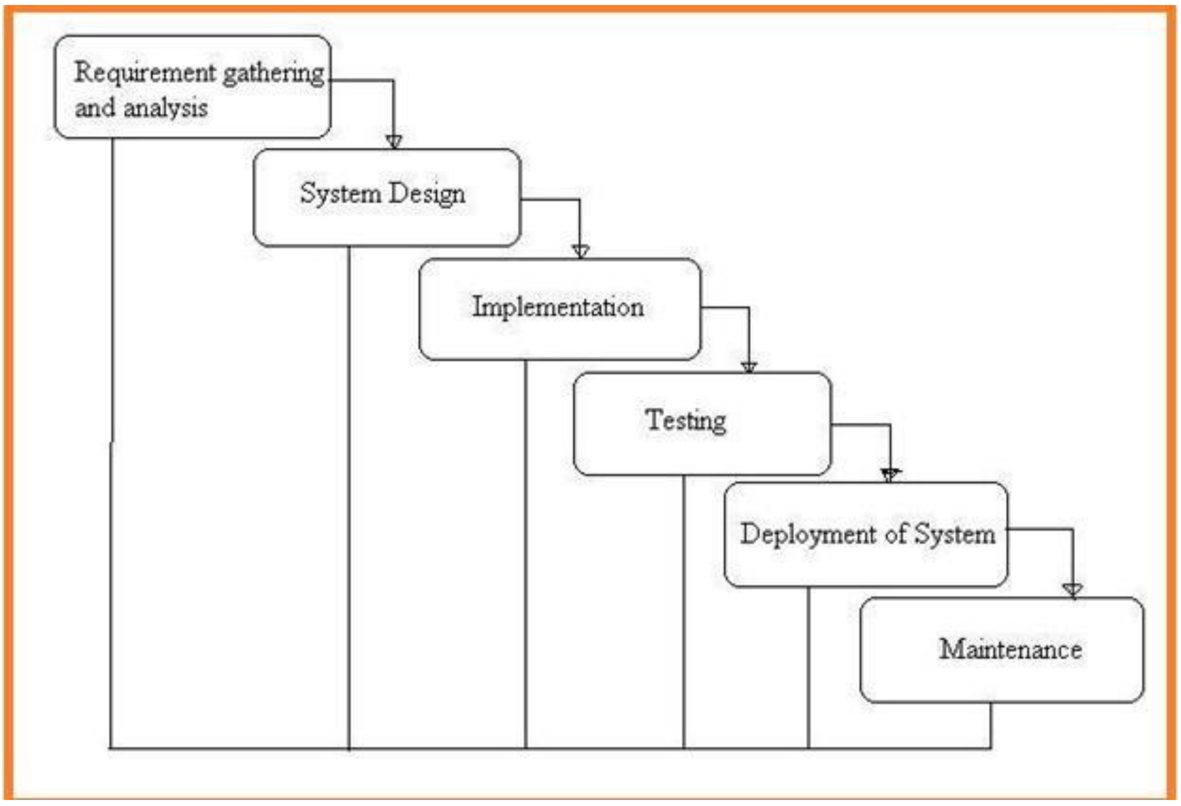


Fig: 1 Project SDLC

- Project Requisites Accumulating and Analysis
- Application System Design
- Practical Implementation
- Manual Testing of My Application
- Application Deployment of System
- Maintenance of the Project

REQUISITES ACCUMULATING AND ANALYSIS

It's the first and foremost stage of the any project as our is a an academic leave for requisites amassing we followed of IEEE Journals and Amassed so many IEEE Relegated papers and final culled a Paper designated "Individual web revisitation by setting and substance importance input and for analysis stage we took referees from the paper and did literature survey of some papers and amassed all the Requisites of the project in this stage

1.6.1 SYSTEM DESIGN

In System Design has divided into three types like GUI Designing, UML Designing with avails in development of project in facile way with different actor and its utilizer case by utilizer case diagram, flow of the project utilizing sequence, Class diagram gives information about different class in the project with methods that have to be utilized in the project if comes to our project our UML Will utilizable in this way The third and post import for the project in system design is Data base design where we endeavor to design data base predicated on the number of modules in our project

IMPLEMENTATION

The Implementation is Phase where we endeavor to give the practical output of the work done in designing stage and most of Coding in Business logic lay coms into action in this stage its main and crucial part of the project

TESTING UNIT TESTING

It is done by the developer itself in every stage of the project and fine-tuning the bug and module predicated additionally done by the developer only here we are going to solve all the runtime errors

MANUAL TESTING

As our Project is academic Leave, we can do any automatic testing so we follow manual testing by endeavor and error methods

DEPLOYMENT OF SYSTEM AND MAINTENANCE

Once the project is total yare, we will come to deployment of client system in genuinely world as its academic leave we did deployment i our college lab only with all need Software's with having Windows OS .

The Maintenance of our Project is one-time process only

2.3 Scope of the Project

With the rapid growth of the highway transportation system, the number of car ownership has risen year after year which is result in serious traffic conditions [1]. In particular, the incidence of curve accidents and the seriousness of accidents remain high. When the car is turning, there will be a blind zone of sight which is accompanied by increased centrifugal force. The turning radius will decrease and the lateral sliding will occur easily, which is caused collision accidents [2]. In Japan, the traffic accident rate on the curved sections of the road exceeded 41.01% of the total accident rate [3], while the number of traffic accidents on the curved road in China accounted for 7.84% of the total accident. Judging from the severity of the accident, the fatal accidents of the curve occupies 16.3% of all fatal accidents [4]. Other statistics show that the main reasons of accidents in the curved areas are the over-speeding of the turning vehicles during turning, irregularly overtaking lane change and lane occupancy [5]. During driving, many accidents occurred due to driver's inattentiveness or unfamiliarity with the road ahead, especially at the curved road which is the place of the high incidence of accidents [6]. Therefore, if it is possible to detect and recognize the road ahead before the advent of curved road conditions, warn the driver in advance, slowdown and avoid evasion in advance, many unnecessary accidents can be avoided and the safety of life and property can be guaranteed.

.

2.4 Modules Description

- Data Acquisition: Upload the URL data from the local host
- Data Preprocessing: In this module, we will perform label encoding, convert the text data into token counts and quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus.
- Spliting: In this module we will split the data into train and test data. x Train and y Train become data for the machine learning, capable to create a model.Once the model is created, input x Test and the output should be equal to y Test. The more closely the model output is to y Test: the more accurate the model is.
- Modelling: in this module, we will apply the CNN-LSTM and CNN-BiLSTM on URL text and we will apply the ,machine learning algorithms on the features of URL.
- Compariosn: Visualize the varies accuracy of modeling
- Prediction: Url phishing detection on the new site

2.5 System Configuration

SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- **Python idel 3.7 version (or)**
- **Anaconda 3.7 (or)**
- **Jupiter (or)**
- **Google colab**

HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

• **Operating system**	**: windows, linux**
• **Processor**	**: minimum intel i3**
• **Ram**	**: minimum 4 gb**
• **Hard disk**	**: minimum 250gb**

3. LITERATURE OVERVIEW

3.1. (2017). Kaspersky Security Bulletin: Overall Statisticals For. Accessed: Jul. 12, 2018. [Online]. Available: https://securelist.com/ksb-overallstatistics- 2017/83453/

The rise of different robotized devices has indicated that the speed with which malware changes on the Internet is far quicker than individuals figured it out. Kaspersky Labs distinguished 15,714,700 malevolent articles in 2017 [1], while the quantity of pernicious documents identified by McAfee Labs expanded to 79 million every day in 2018 (Q1 implies the first quartal), up from 45 million out of 2017 (Q4 implies the fourth quartal) [2]. Despite the fact that the speed with which malware changes on the Internet is getting quicker and quicker, most obscure malware is gotten from known malware. In this way, finding the homology among tests assumes a significant job in following assault sources, reestablishing working conditions, and forestalling assaults. Most malware can be grouped by breaking down static highlights, yet the expansion of the pressing and jumbling methods effectively encourages the formation of malware with predictable conduct and conflicting static highlights. Dynamic examination is required for such malware. Albeit dynamic investigation is compelling in conduct examination, it likewise implies more expense than static examination [3]. In this manner, it is important to locate a powerful mix plan to take care of these issues. That is, static examination can be utilized to order most malware, and dynamic investigation can be completely used to break down the conduct of the malware. In this work, we propose a malware order framework Malscore dependent on likelihood scoring and AI. We initially produce grayscale pictures from crude malware as static highlights and concentrate local API call successions by executing malware in the sandbox as unique highlights. Grayscale pictures can mirror the general framework and static structure of malware, and have been exhibited to be a successful static component [4]. Programming interface call successions with rich semantics and not effectively changed with obscurity procedures are the most generally utilized powerful highlights for mining malware conduct. At that point, we train the classifier S and the classifier D utilizing grayscale pictures and local API call successions, individually. The classifier S depends on Convolutional Neural Networks (CNNs) with Spatial Pyramid Pooling (SPP) [5], and the classifier D depends on factor n−grams(n = 2, 3, 4) and AI. To link the classifier S and the classifier D, this paper proposes a sort of likelihood scoring with likelihood limit (PT). We set the yield of the softmax layer in the classifier S as the likelihood esteem vector, and judge the validity of the grouping result by looking at the likelihood esteem vector of each malware test with PT. The classifier D will additionally break down malware with lower believability. Malscore for the most part uses the recognition speed and minimal effort of static investigation, and the better strength of dynamic examination for stuffed/muddled malware. Pressing/muddling malware will get a likelihood esteem vector after it is broke down by the classifier S. Such malware is scattered and won't be totally like any family. However, such malware can without much of a stretch be sifted through and gone into classifier D. Along these lines, Malscore improves the heartiness for stuffed/muddled malware, and therefore improves the characterization precision. In light of the significant expense of dynamic examination, if all malware tests are contribution to classifier D, this will incredibly expand the discovery cost. We use likelihood scoring to sift through most malware that get dependable order brings about classifier S, and just information temperamental malware into classifier D. Through this technique, the execution times of dynamic investigation is decreased, and the recognition cost of Malscore is diminished. The commitments of this paper are four-overlap: 1) We take grayscale pictures as static highlights and group malware utilizing CNN with SPP layer. Along these lines, Malscore can diminish the data loss of malware brought about by the picture preprocessing. 2) We utilize variable n-grams as unique highlights and apply DF.IDF to choose significant highlights. Therefore, Malscore can bolster relationship between highlights so the semantic data in the malware is held however much as could reasonably be expected. 3) We propose a likelihood scoring to connect static investigation and dynamic examination, which can improve the order precision of Malscore and diminish the arrangement cost of dynamic examination in Malscore. 4) We play out a progression of assessment tests for Malscore on a genuine and huge malware informational collection. The outcomes show that Malscore has higher exactness and lower order cost. The rest of the paper is sorted out as follows. Segment II studies the key research substance of the related work. Area III depicts the framework structure of Malscore. Segments IV and V clarify the procedure that grayscale pictures are investigated by utilizing CNN with SPP, and local API call successions are dissected by utilizing variable n-grams and AI

3.2 A. Ahmad Y, M. Selvakumar, A. Mohammed, A. Mohammed and A. S. Samer, "TrustQR: A New Technique for the Detection of Phishing Attacks on QR Code," Adv. Sci. Lett., vol. 22, no. 10, pp. 2905-2909, Oct. 2016

Graphic black and white squares, known as Quick Response (QR) code is a matrix barcode, which allows easy interaction between mobile and websites or printed material by getting rid of the need of physically composing a URL or contact data. From the pages of magazines to the sides of transports and announcements, QR code innovation is being utilized progressively in cell phones. Lamentably, Phishers have begun utilizing QR code for phishing assaults by utilizing a few highlights of QR code. This paper presents another methodology called "TrustQR" which identifies URL phishing on QR code. It uses QR code specific features and URL features to detect if the QR code content has a phishing URL. Some of the QR code specific features use QR code content and its characteristics like length, type, and level of error correction to generate the cryptography key. This technique uses the machine learning classification technique.

4. System Design

4.1 System Architecture

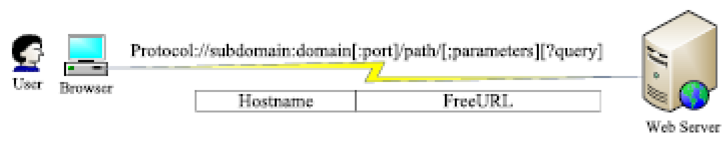

Protocol://subdomain:domain[:port]/path/[;parameters][?query]
User
Browser
Hostname
FreeURL
Web Server

FIGURE 1. The typical structure of a URL.

4.2UML Diagrams

UML DIAGRAMS

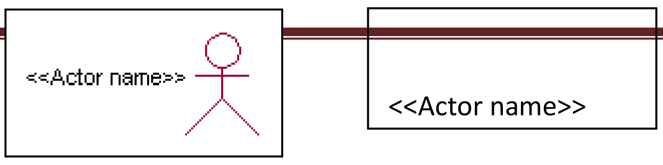
The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

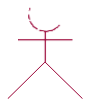
Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:





Actor

An actor is someone or something that:

Interacts with or uses the system.

Provides input to and receives information from the system.

Is external to the system and has no control over the use cases. Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system. Questions to identify actors:
  - Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?

- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

a. System Administrator

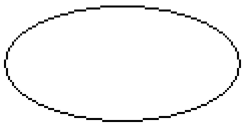
b. Customer

c. Customer Care

Identification of usecases:

Usecase: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor Use cases provide a means to:
- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar. This makes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What usecases will support and maintains the system?

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows Construction of Usecase diagrams:

Use-case diagrams graphically depict system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- actors ("things" outside the system)

- use cases (system boundaries identifying what the system should do)
- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Relationships in use cases:

1. Communication:

The communication relationship of an actor in a usecase is shown by connecting the actor symbol to the usecase symbol with a solid path. The actor is said to communicate with the usecase.

2. Uses:

A Uses relationship between the usecases is shown by generalization arrow from the usecase.

3. Extends:

The extend relationship is used when we have one usecase that is similar to another usecase but does a bit more. In essence it is like subclass.

SEQUENCE DIAGRAMS

A sequence diagram is a graphical view of a scenario that shows object interaction in a time- based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Object:

An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

The object icon is similar to a class icon except that the name is underlined: An object's concurrency is defined by the concurrency of its class.

Message:

A message is the communication carried between two objects that trigger an event. A message carries information from the source focus of control to the destination focus of control. The synchronization of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results.

Link:

A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

CLASS DIAGRAM:

Identification of analysis classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items. There are 4 approaches for identifying classes:

a. Noun phrase approach:
b. Common class pattern approach.
c. Use case Driven Sequence or Collaboration approach.
d. Classes , Responsibilities and collaborators Approach

1. Noun Phrase Approach:

The guidelines for identifying the classes:

- Look for nouns and noun phrases in the usecases.

- Some classes are implicit or taken from general knowledge.
- All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
- Carefully choose and define the class names After identifying the classes we have to eliminate the following types of classes:
- Adjective classes.

2. Common class pattern approach:

The following are the patterns for finding the candidate classes:

- Concept class.
- Events class.
- Organization class
- Peoples class
- Places class
- Tangible things and devices class.

3. Use case driven approach:

We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

4. CRC approach:

The process consists of the following steps:

- Identify classes' responsibilities (and identify the classes)
- Assign the responsibilities
- Identify the collaborators. Identification of responsibilities of each class:

The questions that should be answered to identify the attributes and methods of a class respectively are:

a. What information about an object should we keep track of?

b. What services must a class provide? Identification of relationships among the classes:

Three types of relationships among the objects are:

Association: How objects are associated?

Super-sub structure: How are objects organized into super classes and sub classes? Aggregation: What is the composition of the complex classes?

Association:

The questions that will help us to identify the associations are:

a. Is the class capable of fulfilling the required task by itself?

b. If not, what does it need?

c. From what other classes can it acquire what it needs? Guidelines for identifying the tentative associations:

- A dependency between two or more classes may be an association. Association often corresponds to a verb or prepositional phrase.

- A reference from one class to another is an association. Some associations are implicit or taken from general knowledge.

Some common association patterns are:

Location association like part of, next to, contained in….. Communication association like talk to, order to ……

We have to eliminate the unnecessary association like implementation associations, ternary or n- ary associations and derived associations.

Super-sub class relationships:

Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class).This is based on inheritance.

Guidelines for identifying the super-sub relationship, a generalization are

1. Top-down*:*

Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behavior.

2. Bottom-up*:*

Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

3. Reusability*:*

Move the attributes and methods as high as possible in the hierarchy.

4. Multiple inheritances*:*

Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as an attribute.

Aggregation or a-part-of relationship:

It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn't behave like its parts. It behaves very difficultly. The major properties of this relationship are transitivity and anti symmetry.

The questions whose answers will determine the distinction between the part and whole relationships are:

- Does the part class belong to the problem domain?
- Is the part class within the system's responsibilities?
- Does the part class capture more than a single value?(If not then simply include it as an attribute of the whole class)
- Does it provide a useful abstraction in dealing with the problem domain? There are three types of aggregation relationships. They are:

Assembly:

It is constructed from its parts and an assembly-part situation physically exists.

Container:

A physical whole encompasses but is not constructed from physical parts.

Collection member:

A conceptual whole encompasses parts that may be physical or conceptual. The container and collection are represented by hollow diamonds but composition is represented by solid diamond.

USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

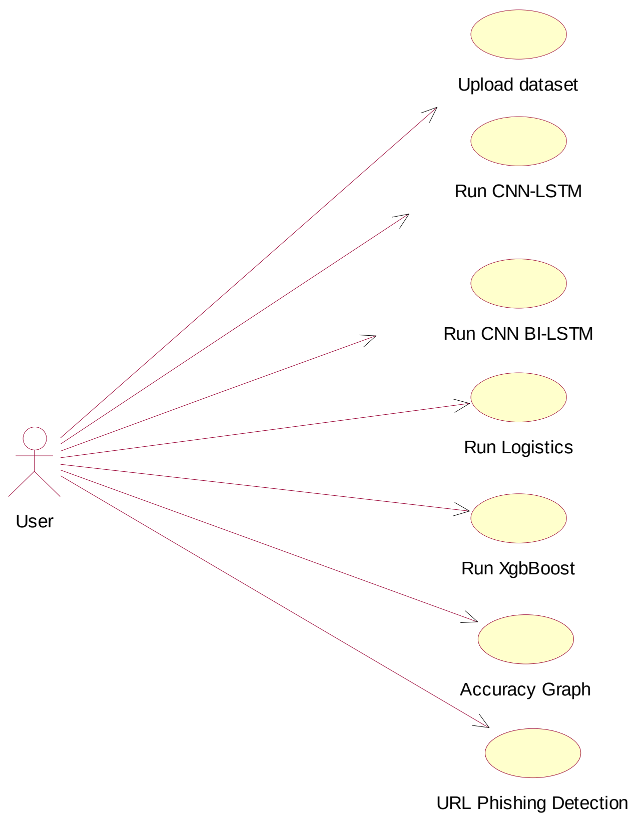


Fig 1: Use Case Diagram

CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

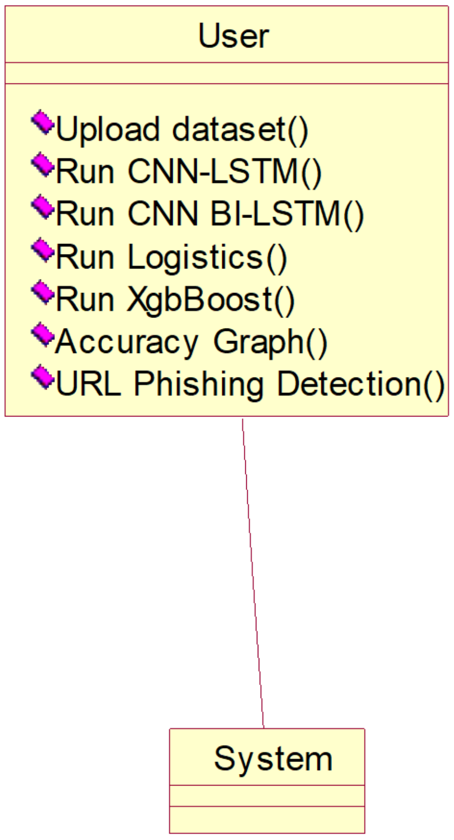


Fig 2:Class Diagram

SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

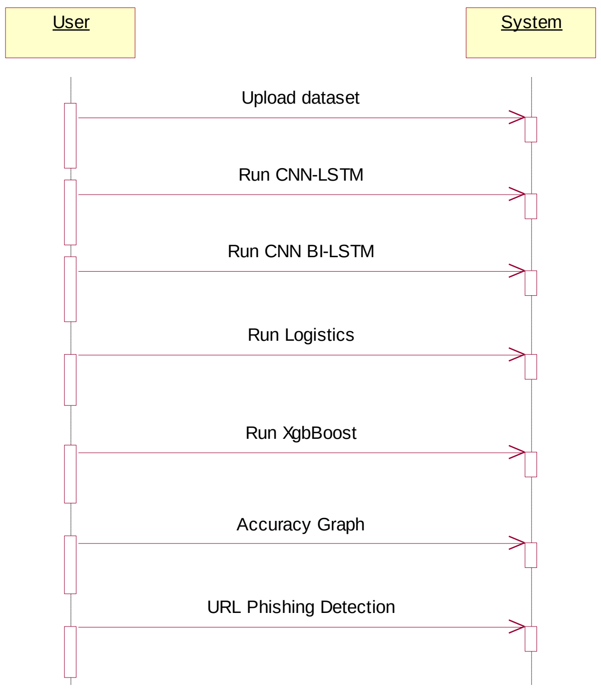
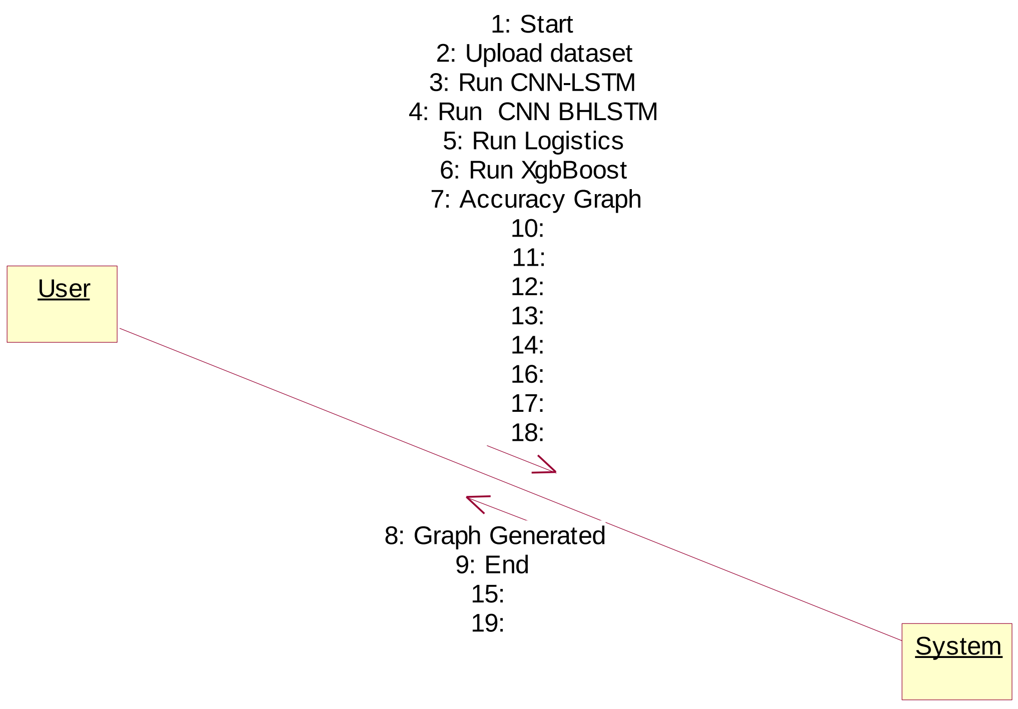
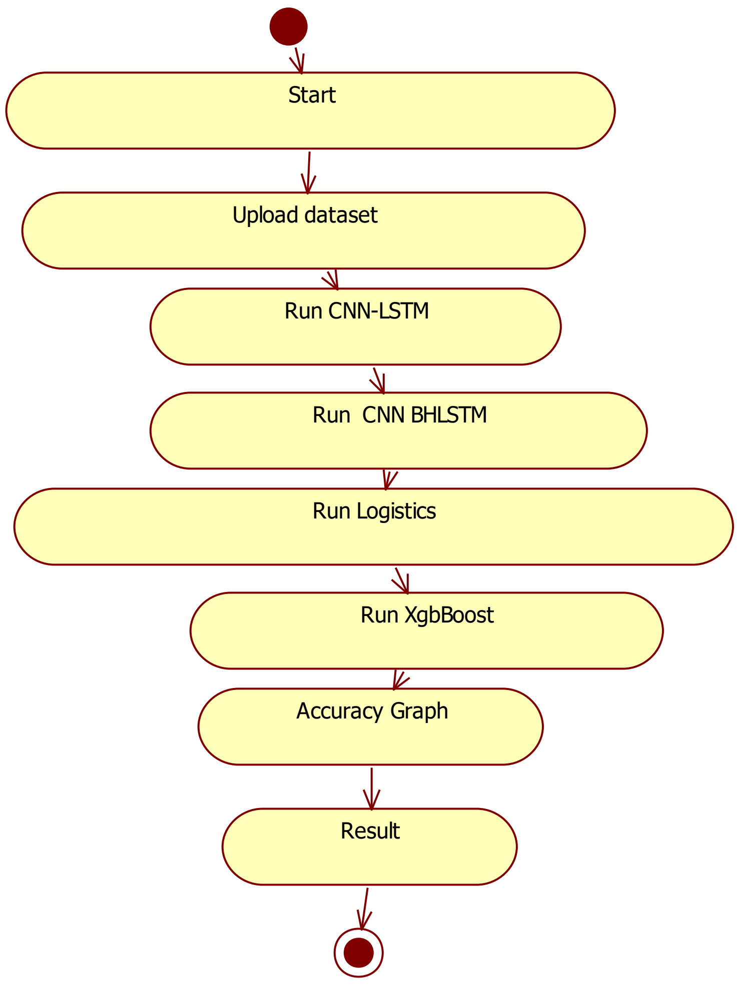


Fig 3: Sequence Diagram

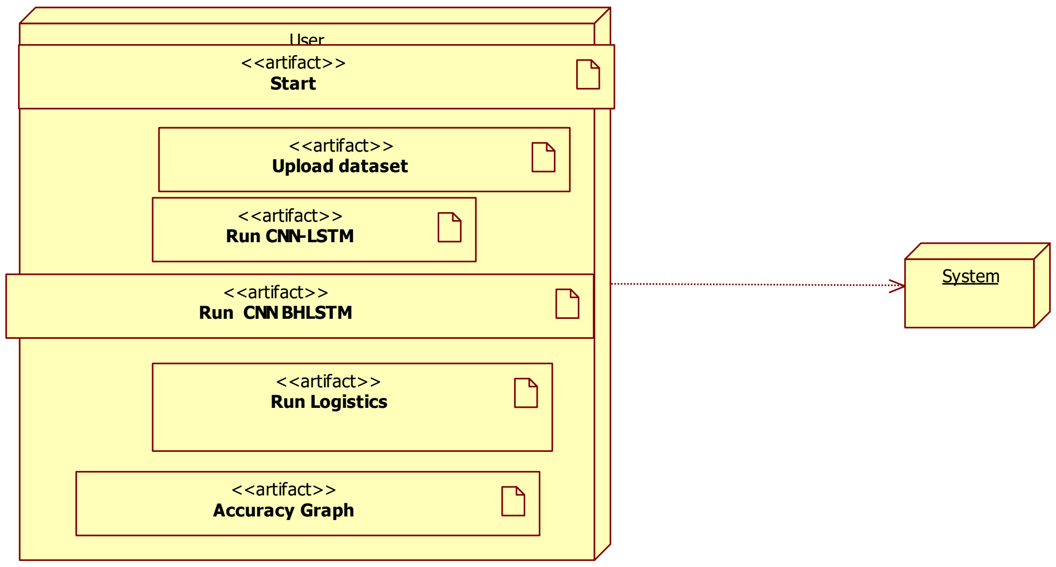
Collaboration Diagram



Activity Diagram



Deployment Diagram



4.3 System Design

5. Sample Code

5.1 Coding

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

from tkinter.filedialog import askopenfilename

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

import numpy as np

import pandas as pd

from genetic_selection import GeneticSelectionCV

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from sklearn import svm

from keras.models import Sequential
from keras.layers import Dense

import time



main = tkinter.Tk()

main.title("Android Malware Detection")

main.geometry("1300x1200")



global filename

global train

global svm_acc, nn_acc, svmga_acc, annga_acc

global X_train, X_test, y_train, y_test

global svmga_classifier

global nnga_classifier

global svm_time,svmga_time,nn_time,nnga_time




def upload():

    global filename

    filename = filedialog.askopenfilename(initialdir="dataset")

    pathlabel.config(text=filename)

    text.delete('1.0', END)

    text.insert(END,filename+" loaded\n");
```

```
def generateModel():
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    train = pd.read_csv(filename)
    rows = train.shape[0]  # gives number of row count
    cols = train.shape[1]  # gives number of col count
    features = cols - 1
    print(features)
    X = train.values[:, 0:features]
    Y = train.values[:, features]
    print(Y)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 
0.2, random_state = 0)



    text.insert(END,"Dataset Length : "+str(len(X))+"\n");
    text.insert(END,"Splitted Training Length : 
"+str(len(X_train))+"\n");
    text.insert(END,"Splitted Test Length : "+str(len(X_test))+"\n\n");



def prediction(X_test, cls):  #prediction done here
```

```
    y_pred = cls.predict(X_test)

    for i in range(len(X_test)):

      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))

    return y_pred



# Function to calculate accuracy

def cal_accuracy(y_test, y_pred, details):

    cm = confusion_matrix(y_test, y_pred)

    accuracy = accuracy_score(y_test,y_pred)*100

    text.insert(END,details+"\n\n")

    text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")

    text.insert(END,"Report : "+str(classification_report(y_test,
y_pred))+"\n")

    text.insert(END,"Confusion Matrix : "+str(cm)+"\n\n\n\n\n")

    return accuracy



def runSVM():

    global svm_acc

    global svm_time

    start_time = time.time()

    text.delete('1.0', END)

    cls = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state =
2)
```

```
    cls.fit(X_train, y_train)

    prediction_data = prediction(X_test, cls)

    svm_acc = cal_accuracy(y_test, prediction_data,'SVM Accuracy')

    svm_time = (time.time() - start_time)


def runSVMGenetic():

    text.delete('1.0', END)

    global svmga_acc

    global svmga_classifier

    global svmga_time

    estimator = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf',
random_state = 2)

    svmga_classifier = GeneticSelectionCV(estimator,

                              cv=5,

                              verbose=1,

                              scoring="accuracy",

                              max_features=5,

                              n_population=50,

                              crossover_proba=0.5,

                              mutation_proba=0.2,

                              n_generations=40,

                              crossover_independent_proba=0.5,
                        mutation_independent_proba=0.05,
                        tournament_size=3,
                        n_gen_no_change=10,
                        caching=True,
                        n_jobs=-1)
    start_time = time.time()
    svmga_classifier = svmga_classifier.fit(X_train, y_train)
    svmga_time = svm_time/2
    prediction_data = prediction(X_test, svmga_classifier)
    svmga_acc = cal_accuracy(y_test, prediction_data,'SVM with GA
Algorithm Accuracy, Classification Report & Confusion Matrix')



def runNN():
    global nn_acc
    global nn_time
    text.delete('1.0', END)
    start_time = time.time()
    model = Sequential()
    model.add(Dense(4, input_dim=215, activation='relu'))
    model.add(Dense(215, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=50, batch_size=64)
    _, ann_acc = model.evaluate(X_test, y_test)
    nn_acc = ann_acc*100
    text.insert(END,"ANN Accuracy : "+str(nn_acc)+"\n\n")
    nn_time = (time.time() - start_time)

def runNNGenetic():
    global annga_acc
    global nnga_time
    text.delete('1.0', END)
    train = pd.read_csv(filename)
    rows = train.shape[0]  # gives number of row count
    cols = train.shape[1]  # gives number of col count
    features = cols - 1
    print(features)
    X = train.values[:, 0:100]
    Y = train.values[:, features]
    print(Y)
    X_train1, X_test1, y_train1, y_test1 = train_test_split(X, Y, test_size
= 0.2, random_state = 0)
    model = Sequential()
    model.add(Dense(4, input_dim=100, activation='relu'))

    model.add(Dense(100, activation='relu'))

    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

    start_time = time.time()

    model.fit(X_train1, y_train1)

    nnga_time = (time.time() - start_time)

    _, ann_acc = model.evaluate(X_test1, y_test1)

    annga_acc = ann_acc*100

    text.insert(END,"ANN with Genetic Algorithm Accuracy :
"+str(annga_acc)+"\n\n")


def graph():

    height = [svm_acc, nn_acc, svmga_acc, annga_acc]

    bars = ('SVM Accuracy','NN Accuracy','SVM Genetic Acc','NN
Genetic Acc')

    y_pos = np.arange(len(bars))

    plt.bar(y_pos, height)

    plt.xticks(y_pos, bars)

    plt.show()
```

```
def timeGraph():
    height = [svm_time,svmga_time,nn_time,nnga_time]
    bars = ('SVM Time','SVM Genetic Time','NN Time','NN Genetic Time')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()



font = ('times', 16, 'bold')
title = Label(main, text='Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning')
#title.config(bg='brown', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 14, 'bold')
uploadButton = Button(main, text="Upload Android Malware Dataset", command=upload)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=460,y=100)

generateButton = Button(main, text="Generate Train & Test Model",
command=generateModel)
generateButton.place(x=50,y=150)
generateButton.config(font=font1)

svmButton = Button(main, text="Run SVM Algorithm",
command=runSVM)
svmButton.place(x=330,y=150)
svmButton.config(font=font1)

svmgaButton = Button(main, text="Run SVM with Genetic
Algorithm", command=runSVMGenetic)
svmgaButton.place(x=540,y=150)
svmgaButton.config(font=font1)
```

```
nnButton = Button(main, text="Run Neural Network Algorithm",
command=runNN)

nnButton.place(x=870,y=150)

nnButton.config(font=font1)


nngaButton = Button(main, text="Run Neural Network with Genetic
Algorithm", command=runNNGenetic)

nngaButton.place(x=50,y=200)

nngaButton.config(font=font1)


graphButton = Button(main, text="Accuracy Graph",
command=graph)

graphButton.place(x=460,y=200)

graphButton.config(font=font1)


exitButton = Button(main, text="Execution Time Graph",
command=timeGraph)

exitButton.place(x=650,y=200)

exitButton.config(font=font1)




font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)
```

```
scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

#main.config()

main.mainloop()
```

6.TESTING

6.1 SOFTWARE TESTING

Testing

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

6.1.1 Types of Testing

1. White Box Testing
2. Black Box Testing
3. Unit testing
4. Integration Testing
5. Alpha Testing
6. Beta Testing
7. Performance Testing and so on

White Box Testing

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers

Black Box Testing

A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

Unit Testing

Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

Integration Testing

The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

Alpha Testing

Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

Beta Testing

Final testing before releasing application for commercial purpose. It is typically done by end- users or others.

Performance Testing

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

Black Box Testing

Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures etc. Test cases for black box testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. Fig.4.1 represents the black box testing:

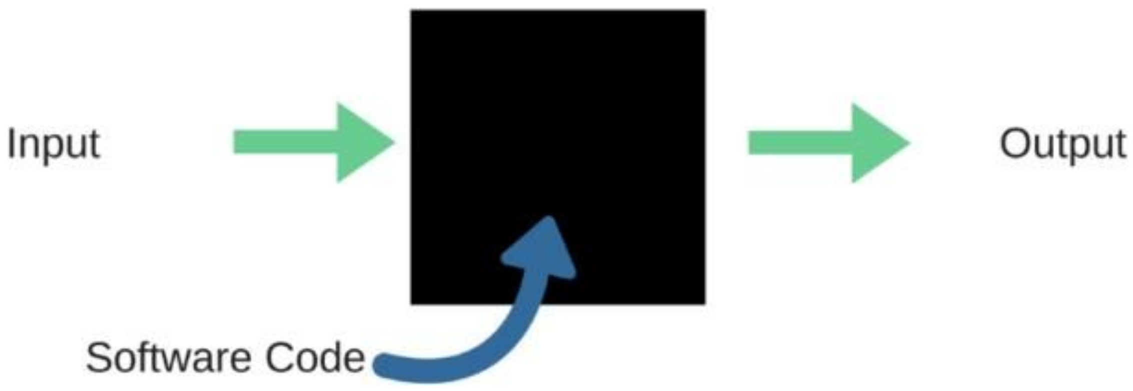


Fig.: Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning

model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.

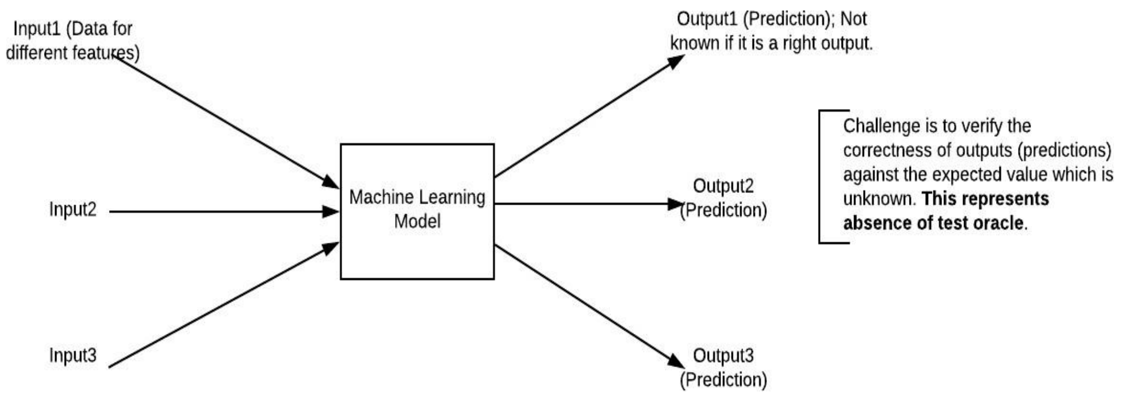

Input1 (Data for different features)
Input2
Input3
Machine Learning Model
Output1 (Prediction); Not known if it is a right output.
Output2 (Prediction)
Output3 (Prediction)
Challenge is to verify the correctness of outputs (predictions) against the expected value which is unknown. **This represents absence of test oracle**.

Fig.:**Black Box Testing for Machine Learning algorithms**

The above Fig.4.2 represents the black box testing procedure for machine learning algorithms.

Table.4.1: Black box Testing

Input	**Actual Output**	**Predicted Output**
[16,6,324,0,0,0,22,0,0,0,0,0,0]	0	0
[16,7,263,7,0,2,700,9,10,1153,832,9,2]	1	1

The model gives out the correct output when different inputs are given which are mentioned in Table 4.1. Therefore the program is said to be executed as expected or correct program

Testing

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

7.2.2 Types of Testing

1. White Box Testing
2. Black Box Testing

3. Unit testing
4. Integration Testing
5. Alpha Testing
6. Beta Testing
7. Performance Testing and so on

White Box Testing

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers

Black Box Testing

A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

Unit Testing

Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

Integration Testing

The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

Alpha Testing

Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

Beta Testing

Final testing before releasing application for commercial purpose. It is typically done by end- users or others.

Performance Testing

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

Black Box Testing

Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures etc. Test cases for black box testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. Fig.4.1 represents the black box testing:

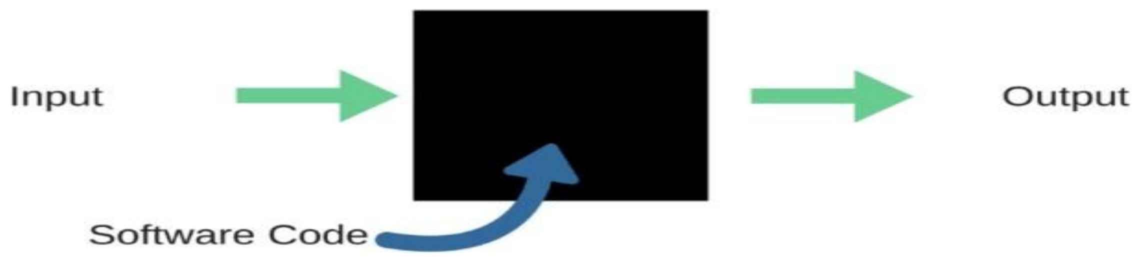


Fig.: Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning

model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.

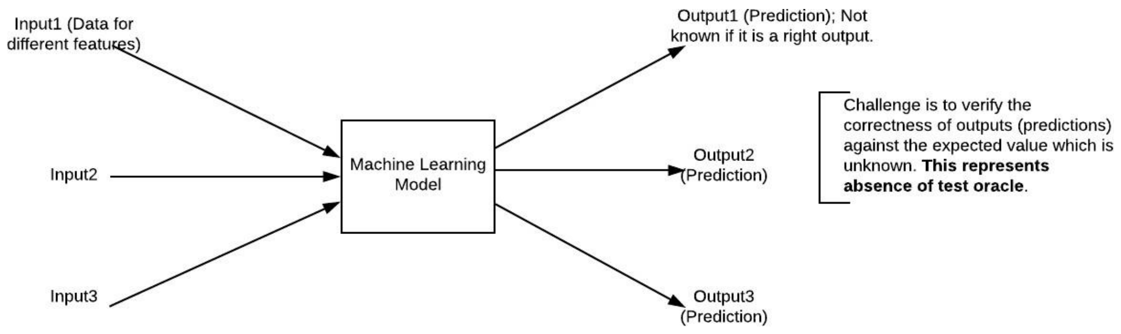

Input1 (Data for different features)
Input2
Input3
Machine Learning Model
Output1 (Prediction); Not known if it is a right output.
Output2 (Prediction)
Output3 (Prediction)
Challenge is to verify the correctness of outputs (predictions) against the expected value which is unknown. This represents absence of test oracle.

Fig.:**Black Box Testing for Machine Learning algorithms**

The above Fig.4.2 represents the black box testing procedure for machine learning algorithms.

Table.4.1: Black box Testing

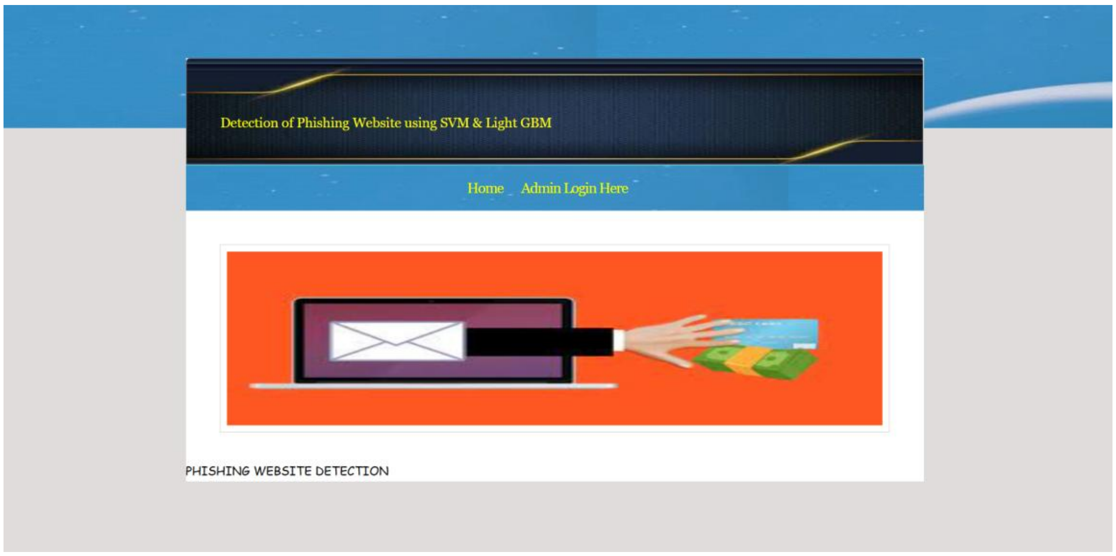
Input	**Actual Output**	**Predicted Output**
[16,6,324,0,0,0,22,0,0,0,0,0,0]	0	0
[16,7,263,7,0,2,700,9,10,1153,832,9,2]	1	1

The model gives out the correct output when different inputs are given which are mentioned in Table 4.1. Therefore the program is said to be executed as expected or correct program

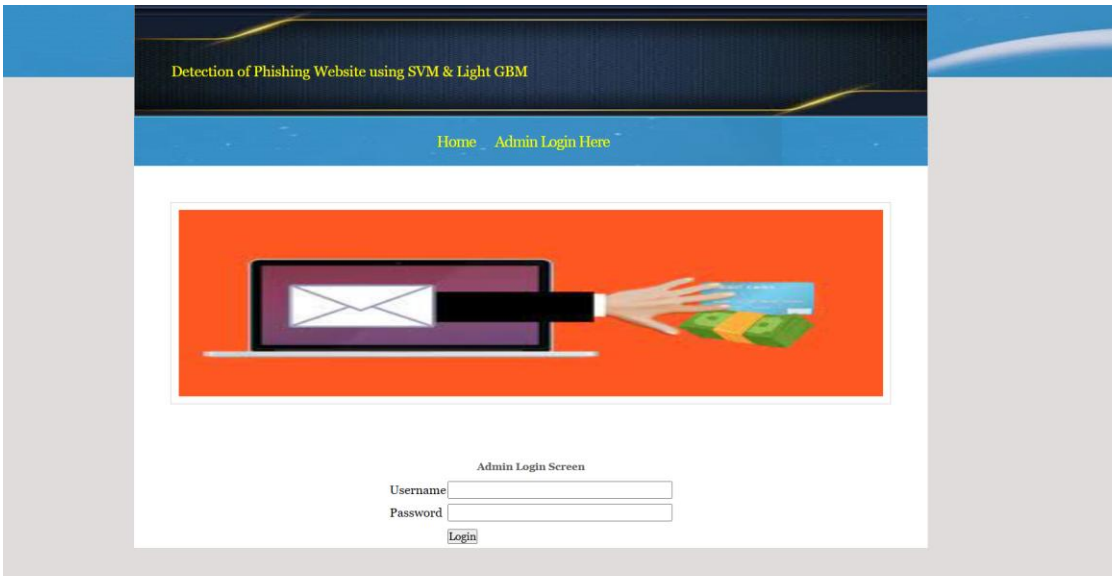
Test Cas e Id	Test Case Name	Test Case Description	Test Steps			Test Case Statu s	Test Priorit Y
			Step	Expected	Actual		
01	Start the Applicatio	Host the application	If it doesn't	We cannot	The application	High	High

	N	and test if it starts making sure the required software is available	Start	run the applicati on.	hosts success.		
02	Home Page	Check the deployment environmen t for properly loading the application.	If it doesn't load.	We cannot access the applicati on.	The application is running successfully .	High	High
03	User Mode	Verify the working of the application in freestyle mode	If it doesn't Respond	We cannot use the Freestyle mode.	The application displays the Freestyle Page	High	High
04	Data Input	Verify if the application takes input and updates	If it fails to take the input or store in The Database	We cannot proceed further	The application updates the input to application	High	High

7.RESULTS AND DISCUSSIONS

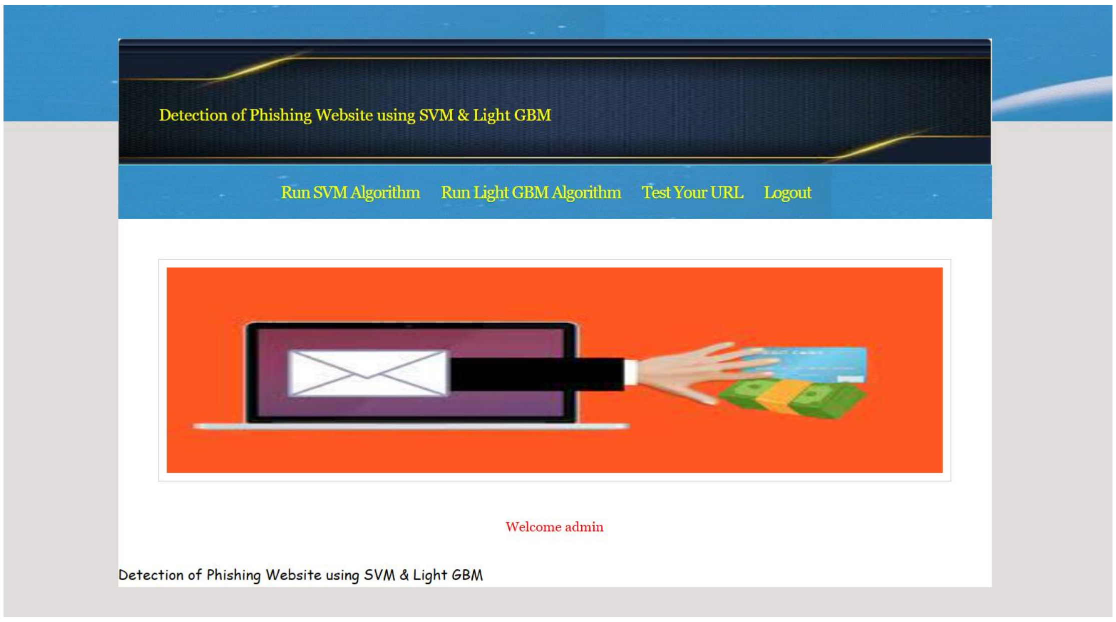

Detection of Phishing Website using SVM & Light GBM
Home
Admin Login Here
PHISHING WEBSITE DETECTION

Now click on the Upload button

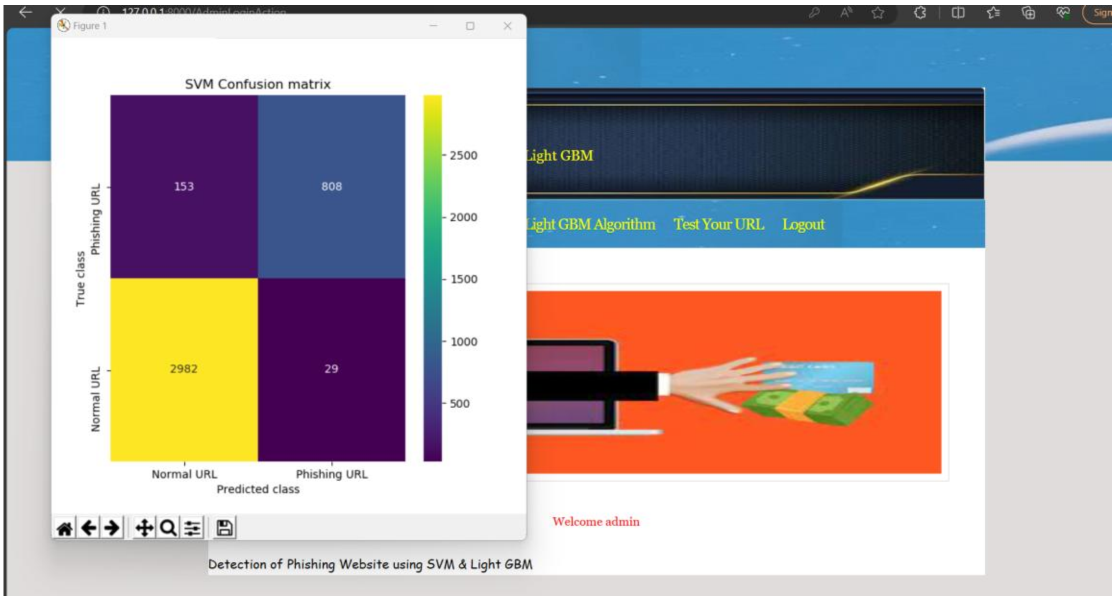


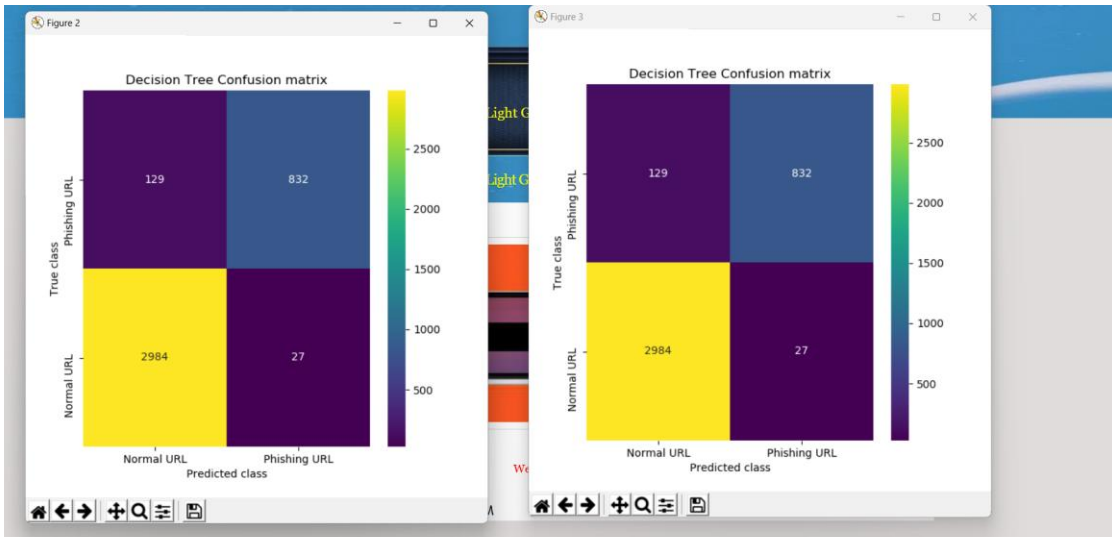
Put user   name- admin

Password – admin


Detection of Phishing Website using SVM & Light GBM
Run SVM Algorithm
Run Light GBM Algorithm
Test Your URL
Logout
Welcome admin
Detection of Phishing Website using SVM & Light GBM

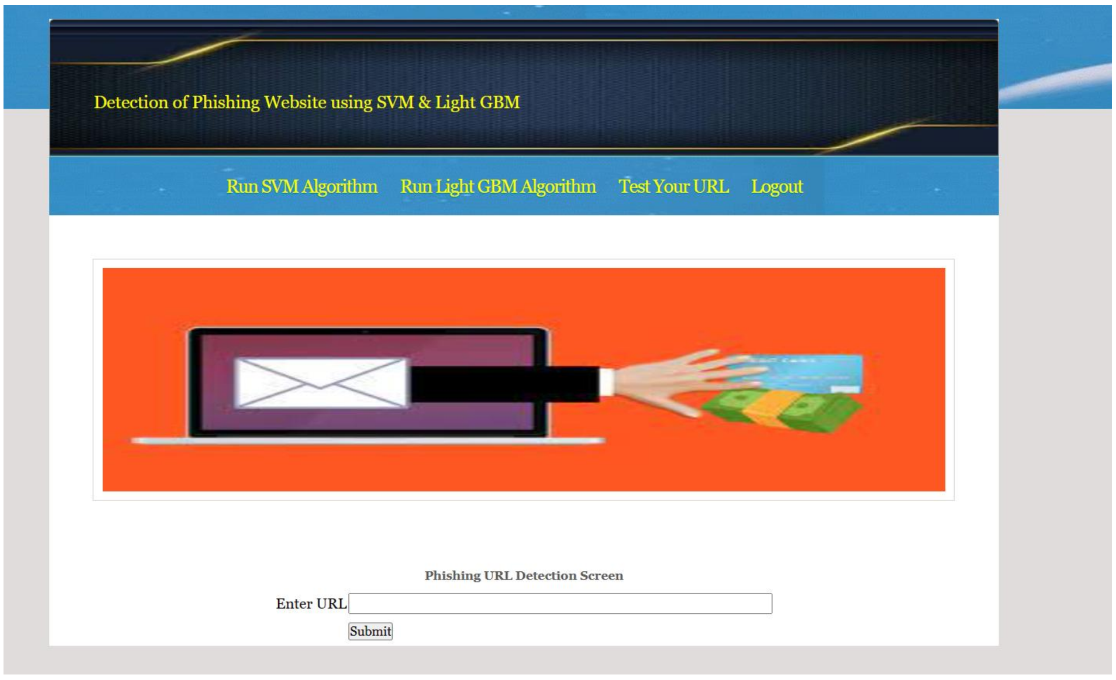
Run svm algorithm





Run light BGM algorithm

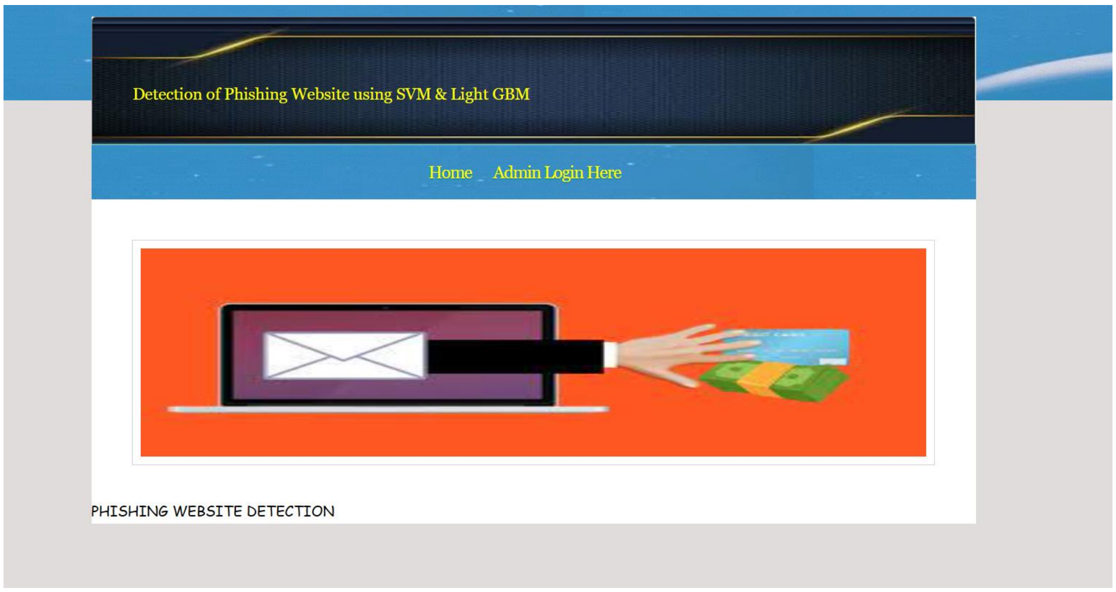
Text your url



Phishing URL Detection Screen

Enter URL

Logout


Detection of Phishing Website using SVM & Light GBM
Home
Admin Login Here
PHISHING WEBSITE DETECTION

1 ➔ URL spam

0 => URL NOT SPAM

CONCLUSION & FUTURE WORK

CONCLUSION

It is well known that a good phishing website detection approach should have good real-time performance while ensuring good accuracy and a low false positive rate. Our proposed MFPD approach is consistent with this idea. Under the control of a dynamic category decision algorithm, the URL character sequence without phishing prior knowledge ensures the detection speed, and the multidimensional feature detection ensures the detection accuracy. We conduct a series of experiments on a dataset containing millions of phishing and legitimate URLs. From the results, we _nd that the MFPD approach is effective with high accuracy, low false positive rate and high detection speed. A future development of our approach will consider applying deep learning to feature extraction of webpage code and webpage text. In addition, we plan to implement our approach into a plugin for embedding in a Web browser.

FUTURE SCOPE

However, these work mainly focus on generic data, and it is not clear whether they can be applied to text data. With the proliferation of text applications, we may see a trend that these feature selection techniques will be applied to text categorization, and interesting problems may arise, for example,feature selection for text categorization when there are missing values in documents [116].

REFERENCES

[1] (2018). *Phishing Attack Trends Re-Port-1Q*. Accessed: May 5, 2018.

[Online]. Available: https://apwg.org/resources/apwg-reports/

[2] (2017). *Kaspersky Security Bulletin: Overall Statisticals For*. Accessed:

Jul. 12, 2018. [Online]. Available: https://securelist.com/ksb-overallstatistics-

2017/83453/

[3] A.Y. Ahmad, M. Selvakumar, A. Mohammed, and A.-S. Samer, ``TrustQR:

A new technique for the detection of phishing attacks on QR code," *Adv.*

Sci. Lett., vol. 22, no. 10, pp. 2905_2909, Oct. 2016.

[4] C. C. Inez and F. Baruch, ``Setting priorities in behavioral interventions:

An application to reducing phishing risk," *Risk Anal.*, vol. 38, no. 4,

pp. 826_838, Apr. 2018.

[5] G. Diksha and J. A. Kumar, ``Mobile phishing attacks and defence mechanisms:

State of art and open research challenges," *Comput. Secur.*, vol. 73,

pp. 519_544, Mar. 2018.

[6] *Google Safe Browsing APIs*. Accessed: Oct. 1, 2018. [Online]. Available:

https://developers.google.com/safe-browsing/v4/

[7] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, ``An empirical analysis of phishing blacklists," in *Proc. 6th Conf. Email Anti-Spam (CEAS)*, Jul. 2009, pp. 59_78.

[8] A. K. Jain and B. B. Gupta, ``A novel approach to protect against phishing attacks at client side using auto-updated white-list," *EURASIP J. Inf. Secur.*, vol. 2016, no. 1, Dec. 2016, Art. no. 34.

[9] M. Zouina and B. Outtaj, ``A novel lightweight URL phishing detection system using SVM and similarity index," *Hum.-Centric Comput. Inf. Sci.*, vol. 7, no. 1, p. 17, Jun. 2017.

[10] E. Buber, Ö. Demir, and O. K. Sahingoz, ``Feature selections for the machine learning based detection of phishing websites," in *Proc. IEEE Int. Artif. Intell. Data Process. Symp. (IDAP)*, Sep. 2017, pp. 1_5.

[11] J. Mao *et al.*, ``Detecting phishing websites via aggregation analysis of page layouts," *Procedia Comput. Sci.*, vol. 129, pp. 224_230, Jan. 2018.

[12] J. Mao,W. Tian, P. Li, T.Wei, and Z. Liang, ``Phishing-alarm: Robust and ef_cient phishing detection via page component similarity," *IEEE Access*, vol. 5, no. 99, pp. 17020_17030, Aug. 2017.

[13] J. Cao, D. Dong, B. Mao, and T. Wang, ``Phishing detection method based on URL features," *J. Southeast Univ.-Engl. Ed.*, vol. 29, no. 2, pp. 134_138, Jun. 2013.

[14] S. C. Jeeva and E. B. Rajsingh, ``Phishing URL detection-based feature selection to classi_ers," *Int. J. Electron. Secur. Digit. Forensics*, vol. 9,

no. 2, pp. 116_131, Jan. 2017.

[15] A. Le, A. Markopoulou, and M. Faloutsos, ``PhishDef: URL names say it all,'' in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Sep. 2010, pp. 191_195.

[16] R. Verma and K. Dyer, ``On the character of phishing URLs: Accurate and robust statisticalal learning classi_ers,'' in *Proc. 5th ACMConf. Data Appl. Secur. Priv. (ACM CODASPY)*, Mar. 2015, pp. 111_122.

[17] Y. Li, S. Chu, and R. Xiao, ``A pharming attack hybrid detection model based on IP addresses and Web content,'' *Optik*, vol. 126, no. 2, pp. 234_239, Nov. 2014.

[18] G. G. Xiang and J. Hong, ``A hybrid phish detection approach by identity discovery and keywords retrieval,'' in *Proc. Int. Conf. World Wide Web (WWW)*, Oct. 2009, pp. 571_580

[19] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, ``CANTINAC: A featurerich machine learning framework for detecting phishing Web sites,'' *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, p. 21, Sep. 2011.

[20] S. Marchal, K. Saari, N. Singh, and N. Asokan, ``Know your phish: Novel techniques for detecting phishing sites and their targets,'' in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2016, pp. 323_333.

[21] R. Patil, B. D. Dhamdhere, K. S. Dhonde, and R. G. Chinchwade, ``A hybrid model to detect phishing-sites using clustering and Bayesian approach,'' in *Proc. IEEE Int. Conf. Converg. Technol. (I2CT)*, Apr. 2014, pp. 1_5.

[22] M. Arab and M. K. Sohrabi, ``Proposing a new clustering method to detect phishing websites,'' *Turkish J. Electr. Eng. Comput. Sci.*, vol. 15, no. 1, pp. 92_95, Jun. 2015.

[23] A. Shinde, A. Pandey, R. Pawar, and V. Gangule, ``Clustering and Bayesian approach-based model for detection of phishing,'' *Int. J. Comput. Appl.*, vol. 118, no. 24, pp. 30_33, May 2015.

[24] X. Zhang, Z. Yan, H. Li, and G. Geng, ``Research of phishing detection technology,'' *Chin. J. Netw. Inf. Secur.*, vol. 3, no. 7, pp. 7_24, Jul. 2017.

[25] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, ``Beyond blacklists: Learning to detect malicious web sites from suspicious URLs,'' in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discover Data Mining (KDD)*, Jan. 2009, pp. 1245_1254.

[26] R. M. Mohammad, F. Thabtah, and L. Mccluskey, ``Predicting phishing websites based on self-structuring neural network,'' *Neural Comput. Appl.*, vol. 25, no. 2, pp. 443_458, Aug. 2014.

[27] A. K. Jain and B. B. Gupta, ``Towards detection of phishing websites on client-side using machine learning based approach,'' *Telecommun. Syst.*, vol. 68, no. 4, pp. 687_700, Aug. 2018.

[28] J. Zhang, Y. Ou, D. Li, and Y. Xin, ``A prior-based transfer learning method for the phishing detection,'' *J. Netw.*, vol. 7, no. 8, pp. 1201_1207, Aug. 2012.

[29] L. Chang *et al.*, ``Convolutional neural networks in image understanding,'' *Acta. Automatica Sinica*, vol. 42, no. 9, pp. 1300_1312, Jul. 2016.

[30] Z. C. Lipton, J. Berkowitz, and C. Elkan. (Oct. 2015). ``A critical review of recurrent neural networks for sequence learning." [Online]. Available: https://arxiv.org/abs/1506.00019

[31] S. G. Selvaganapathy, M. Nivaashini, and H. P. Natarajan, ``Deep belief network based detection and categorization of malicious URLs," *Inf. Secur. J., Global Perspective*, vol. 27, no. 3, pp. 145_161, Apr. 2018.

[32] A. C. Bahnse, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. Gonález, ``Classifying phishing URLs using recurrent neural networks," in *Proc. IEEE APWG Symp. Electron. Res. (eCrime)*, Apr. 2017, pp. 1_8.

[33] X. Zhang, J. Zhao, and Y. LeCun, ``Character-level convolutional networks for text classi_cation," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2015, pp. 649_657.