Project Report on

SMART MALICIOUS URL'S DETECTION SYSTEM TO PREVENT PHISHING USING DEEP LEARNING APPROACH

Submitted in partial fulfillment of the requirements

of the degree of Bachelor in Engineering

by

Jainam Soni	BE4- 20
Palak Nisar	BE4- 07
Shamika Dumbre	BE3- 62
Siddhi Sheth	BE4- 17

Under the guidance of

Prof. Deepti Nikumbh

DEPARTMENT OF COMPUTER ENGINEERING

SHAH AND ANCHOR KUTCHHI ENGINEERING COLLEGE

CHEMBUR, MUMBAI- 4000 088.

2018-2019

www.shahandanchor.com Tel: 0222558 0854

Mahavir Education Trust's

SHAH & ANCHOR KUTCHHI ENGINEERING COLLEGE

Mahavir Education Trust Chowk, W.T. Patil Marg, Chembur, Mumbai 400 088

Affiliated to University of Mumbai, Approved by D.T.E. & A.I.C.T.E.

Awarded provisional accreditation for Computer & Electronics Engineering by NBA

(for 2 years from 06-08-2014)

ISO9001:2008 Certififed

Certificate

This is to certify that the report of the project entitled

Smart Malicious Url's Detection System To Prevent Phishing Using Deep Learning Approach

is a bonafide work of

Jainam Soni	BE4- 20
Palak Nisar	BE4- 07
Shamika Dumbre	BE3- 62
Siddhi Sheth	BE4- 17

submitted to the

UNIVERSITY OF MUMBAI

during semester VIII in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER ENGINEERING.

(Prof. Deepti Nikumbh)
Guide

(Prof. Uday Bhave)
I/c Head of Department

(Dr. Bhavesh Patel)
Principal

Approval for Project Report for B. E. Semester VIII

This project report entitled Smart Malicious Url's Detection System To Prevent Phishing Using Deep Learning Approach by Jainam Soni, Palak Nisar, Siddhi Sheth, Shamika Dumbre is approved for semester VIII in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering.

Examiners

1._____

2._____

Guide

1._____

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of student	Roll No.	Signature
Jainam Soni	BE4-20	
Palak Nisar	BE4-07	
Shamika Dumbre	BE3-62	
Siddhi Sheth	BE4-17	

Date:

Attendance Certificate

Date: 05/01/2019

To,
The Principal
Shah and Anchor Kutchhi Engineering College,
Chembur, Mumbai-88

Subject: Confirmation of Attendance

Respected Sir,

This is to certify that Final year (BE) students

Jainam Soni BE4-20
Palak Nisar BE4-07
Shamika Dumbre BE3-62
Siddhi Sheth BE4-17

have duly attended the sessions on the day allotted to them during the period from 07/01/2019 to 20/04/2019 for performing the Project titled Smart Malicious Url's Detection System to prevent Phishing using Deep Learning Approach.

They were punctual and regular in their attendance. Following is the detailed record of the student's attendance.

Attendance Record:

Date	Jainam Soni	Palak Nisar	Shamika Dumbre	Siddhi Sheth
	Present/Absent	Present/Absent	Present/Absent	Present/Absent
09/1/2019	Present	Present	Present	Present
30/1/2019	Present	Present	Absent	Present
18/2/2019	Present	Present	Present	Present
01/3/2019	Present	Present	Present	Present
08/3/2019	Present	Present	Absent	Present
11/3/2019	Present	Present	Present	Present
20/3/2019	Present	Absent	Present	Present
25/3/2019	Present	Present	Present	Absent
27/3/2019	Present	Present	Present	Present
16/4/2019	Present	Present	Present	Present

Signature and Name of Internal Guide

Abstract

Over the past years, there has been an increase in the amount of phishing attacks and security threats. Phishing is a malicious practice in which the attacker fraudulently acquires confidential information like bank details, credit card details, or passwords from legitimate users. In phishing, users are tricked with an phished website containing malicious url's rather than legitimate one. Traditionally, this is done through the usage of blacklists, which cannot be exhaustive, and cannot detect newly generated malicious URLs. To address this, recent years have witnessed several efforts to perform Malicious URL Detection using Machine Learning. Here we propose an anti-phishing technique to safeguard our web experiences. Our approach uses an alternative approach of Feature Deep learning, where an embedding layer takes care of deriving feature vectors from the raw data. These features are passed to Long Short Term Memory (LSTM) to predict whether the url is malicious or benign. The results obtained from our experiment shows that our proposed methodology is very effectual for preventing such attacks rather than other Traditional algorithms and Recurrent Neural Network (RNN) as they suffer from several limitations such as:- (i) Requiring substantial manual feature engineering and (ii) Inability to handle unseen features and generalize to test data.

Keywords: Social Engineering, Blacklists, URL, Phishing, Deep Learning, RNN, ANN, LSTM.

Table of Content

List of Figures

List of Tables

Chapter 1
Introduction

An identity theft that occurs when a malicious web site masquerades a legitimate one is called Phishing. Such a theft occurs in order to procure sensitive information such as passwords, bank account details, or credit card numbers. The makers of such illegitimate website made them exactly look like a legitimate one so that no user can identify the difference easily. The phishing attackers use different kind of social engineering tactics to lure users for example: giving attractive offers to just visit the site. Malicious URL is a URL created with malicious purposes, to download any type of malware to the affected computer, which can be contained in spam or phishing messages, or even improve its position in search engines using Blackhat SEO techniques.

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to learn. Our approach uses an automatic feature extraction of a website to detect any suspicious or phishing website. These features are passed to Long Short Term Memory (LSTM) to predict whether the url is malicious or benign. The results obtained from our experiment shows that our proposed methodology is very effectual for preventing such attacks and the performance was measured by using Confusion Matrix for the classifier.

1.1 Objectives

To improve the generality of malicious URL detectors, machine learning techniques have been explored with increasing attention in recent years. So we here try to develop a system with various machine learning techniques or deep learning approach that gives the best results.

To develop a Smart Malicious URL phishing detection system for end user with the following characteristics:

- Malicious URL Detection System to curb the Phishing attacks.
- The GUI of our system will engage end users and provide user friendly experience.
- System will be based on discerning Url's by their patterns.
- No manual Feature Extraction.
- Maximize the prediction result of the system than the others which are developed.

Create awareness about phishing attack and some other cyber security threats..

1.2 Methodology

The purpose of this project is to build a classifier that can detect malicious URLs. This is accomplished using a Featureless Deep Learning approach. There were different traditional approaches used but instead we use the Deep Learning one. First we collect the Url from different data sources such as different websites and some manual collections. Then we processed the data i.e url to be specific length and padded if short as, url length has maximum feature importance. Now data is trained with Lstm model which uses keras and Tensorflow as backend. This all is done in Google Colaboratory. Parameter tunings are done accordingly to get the best accuracy and for the efficiency of the model. Once the model is prepared it can be used to predict the different urls.

1.3 Organization of Report

The main body of the report is preceded by detailed contents of the report. This is followed by executive summary giving briefly the scope and objectives of the study, importance of the topic, methodology, major observations / findings, and recommendations & action plan.

Chapter 1 Gives an introduction to the project.

Chapter 2 Discusses the research done to tackle the problems faced.

Chapter 3 Gives the problem statement

Chapter 4 Lists the Hardware and Software requirements for the proposed system.

Chapter 5 Explains the design and algorithm for the system.

Chapter 6 Explains all the modules in the system and the screenshots of the UI.

Chapter 7 Discusses the test cases and their results.

Chapter 8 Discusses the results and analyse them.

Chapter 9 Gives the conclusion and future scope of the project.

Chapter 2
Literature Survey

Typically, phishing attacks can be exploited by sending spoofed link to the trusted user and then redirected them to bogus website. Whenever user will enter the important information to that fake website then immediately this information will store to the system of the hacker and then finally the user will be redirected to any other websites that will not be related to the user. There were many researches that were conducted to detect phishing attacks.

Phishtank is a website which identifies if a website is legitimate on the basis of the data stored in it.

The paper describes a supervised machine learning classification model that has been built to detect the distribution of malicious content in online social networks (OSNs). For the data collection stage, the Twitter streaming application programming interface (API) was used and Virus Total was used for labelling the dataset. It uses a Random Forest Classification Model with the combination of different features. This led to a recall value of 0.89 and after parameter tuning and feature selection method, it could improve the performance to 0.92[1].

There is another technique that is developed which focuses on detecting phishing attacks but on mobile phones using anti-phishing scheme on mobile based platforms. It was a challenging task to apply this technique because mobile phone and users have more limitation [6].

An approach was proposed to prevent against the Phishing attacks by the use of auto-updating white lists. In this approach, when user tries to open a website, the browser warns him/her to not access the website if that website is not available in the white-list. This technique also examines the legitimacy of given website using hyperlink features. Hyperlinks are extracted from the source code of given webpage and are passed as an input to the proposed phishing detection algorithm. This proposed approach is effective as it has true positive rate of 86.02 % while false negative rate lesser 1.48% [4].

One the service which the Google provides for safe browsing allows to check the URL against a list of malicious domains that is constantly updated by Google. The Safe Browsing Lookup API permits to pass the suspicious website's URL to Safe Browsing service which tells if the URL sent by the client is benign or malicious. The client URLs are verified using the malicious and phishing lists maintained by Google. However, this approach has the following shortcomings: (i) Before sending the URL hashing is not performed and (ii) There isn't any constraint over the response time taken server to lookup.

Phishing URL detection can be done via proactive or reactive means. On the reactive end, we find services such as Google Safe Browsing API3. This type of services expose a blacklist of malicious URLs to be queried. Blacklists are constructed by using different techniques, including manual reporting, honeypots, or by crawling the web in search of known phishing characteristics. For example, browsers make use of blacklists to block access upon reaching the URLs contained in them. One drawback of such reactive method is that in order for a phishing URL to be blocked, it must be previously included in the blacklist. This implies that web users remain at risk until the URL is submitted and the blacklist is updated. What is more, since the majority of phishing sites are active for less than a day, their mission is complete by the time they are added to the blacklist. Proactive methods mitigate this problem by analyzing the characteristics of a web page in real time in order to assess the potential risk of a web page.

Risk assessment is done through a classification model . Some of the machine learning methods that have been used to detect phishing include: support vector machines , streaming analytics , gradient boosting, random forests and neural networks [6]. Several of these methods employ an array of website characteristics, which mean that in order to evaluate a site, first it has to be rendered before the algorithm can be used. This adds a significant amount of time to the evaluation process. Using URLs, instead of content analysis, reduces the evaluation time because only a limited portion of text is analyzed.

Lately, the application of machine learning techniques for URL classification has been gaining attention. Several studies proposing the use of classification algorithms to detect phishing URLs have come to the light in recent years. These studies are mainly focused on creating features through expert knowledge and lexical analysis of the URL [6]. Then, the phishing site's characteristic are used as quantitative input for the model. The model in turn learns to recognize patterns and associations the inputs must follow in order to label a site as legitimate or malicious.

Now we have also seen that the deep neural networks such Rnn, Lstm, Cnn, etc are used in various applications such as text prediction, language based modelling, classification, etc. Text Generation is one such task which can be be architectured using deep learning models, particularly Recurrent Neural Networks. A trained language model learns the likelihood of occurrence of a word based on the previous sequence of words used in the text. Rnn is used to predict the sentiment classification, text prediction, etc. But the problem was that it can look for references in nearby of 5-10 steps only. While the Lstm can takes the references for the one's which is even 1000 step-sequence behind and so Lstm is more perfect for them. Now as it is used for predicting the text, we can use the Lstm for Domain name generation and use that to check if the domain name i.e. Url is benign or Malicious. Various other techniques have also been emerging to predict the genuinity of the url's.

Chapter 3
Problem Statement

Phishing remains an active threat vector with the volume of attacks growing, according Proofpoint's 2019 State of the Phish report, released on Jan. 24 2019. Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Besides, the most common technique used, blacklist-based method is inefficient in responding to emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database. Furthermore, page content inspection has been used by some strategies to overcome the false negative problems. Moreover, page content inspection algorithms each have different approach to phishing website detection with varying degrees of accuracy. The main purpose of the system is to protect the users and companies from being exposed to malicious content and phishing attacks i.e. to minimize the cyber threats that is spread over internet rapidly and is growing each day with increase in newer technologies and with the large exposure of Internet. So, we deploy a system and a Google Chrome Extension that helps users and companies to be safe from Phished url’s using Deep learning Approach.

.

Chapter 4
System Requirements

4.1 Functional Requirements

- The system should have a simple and easy to user interface with a submit button to proceed url.
- Extension should automatically take the Url from the Search bar.
- Extension should be simple and it should give the Result there itself.

4.2 Hardware Requirements:

- Computer with minimum configuration of processor 1.33 GHz.
- 512mb RAM.
- 80 GB hard disk.

4.3 Software Requirements:

- Python 3.6 or Higher or (Jupyter Notebook).
- Python libraries installed mainly Tensorflow, Keras, Tkinter(For Gui Purpose).
- It can be used as an executable with the help of PyInstaller. (.exe extension file for ease of users.)
- Google Chrome (For Extension Purpose).
- Google Colaboratory (For Training purpose as we require huge computation).

Chapter 5

Project Design

5.1 Proposed Architecture Design.

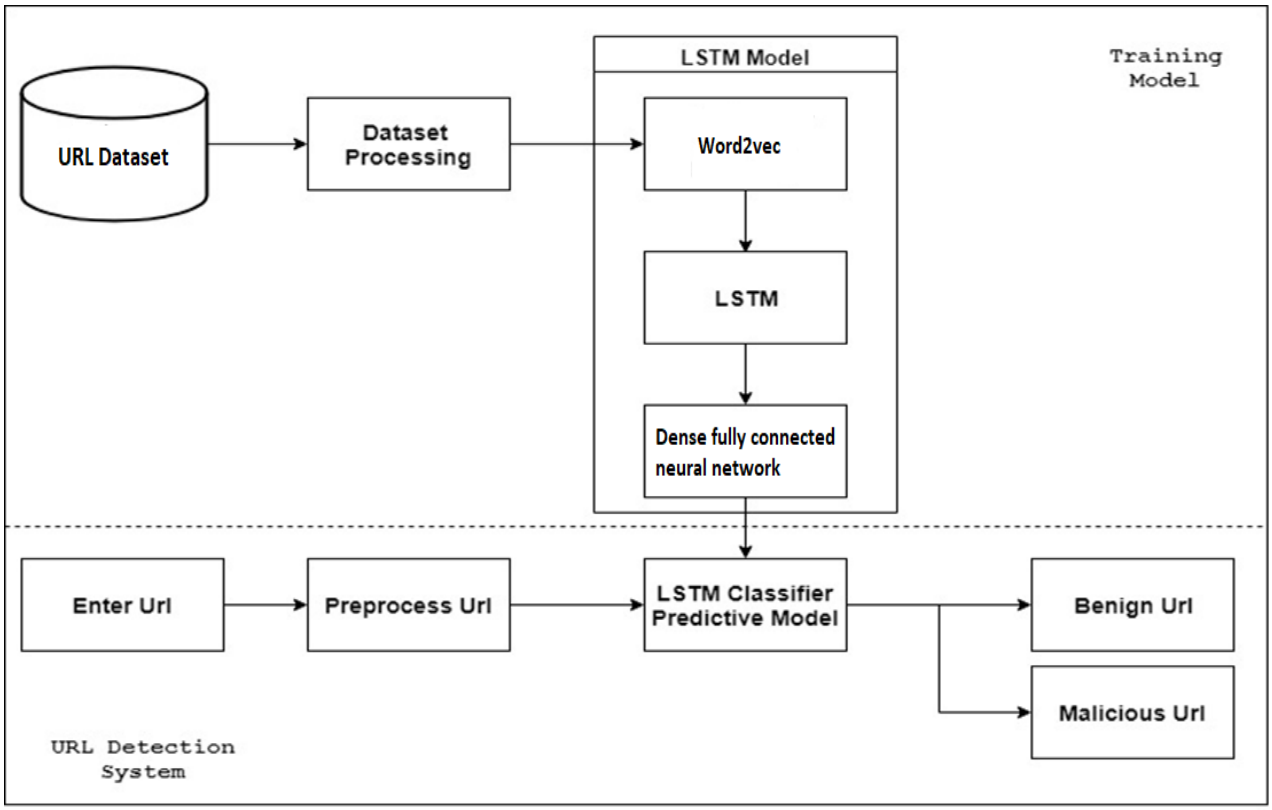
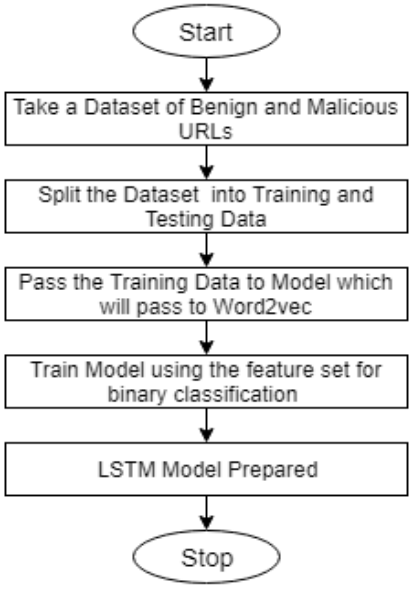


Figure 5.1 Proposed Architecture Design

The above proposed architecture design shows that our project is divided majorly into two parts: i) Training and Testing and ii) Url Detection System which will predict from the Trained Model. Url dataset needs to be processed and then we need to pass the processed Url to Lstm Model built code to successfully build the LSTM Model. Once the model is prepared the url entered by the user is predicted using LSTM classifier predictive model.

5.2 User Interface Design

5.2.1 System Flowchart



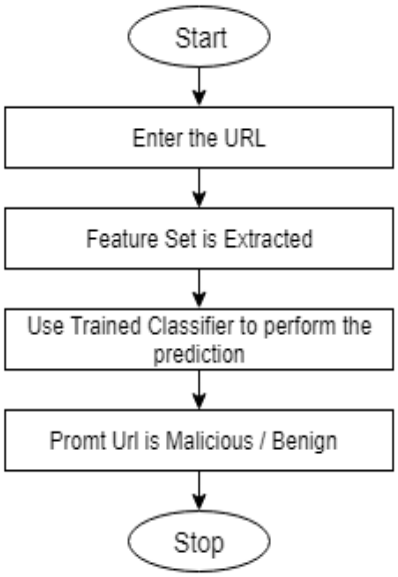


Fig 5.2.1.1 Training flowchart Fig 5.2.1.2 Smart url detection system

5.3 Data

Data is distinct pieces of facts, statistics and information that is collected together and usually formatted in a special way for reference or analysis. Our mechanism uses the Uniform Resource Locator (URL) itself as data without accessing the content of Websites and analyzes it. We have used two types of urls in our dataset which are urls of malicious sites and urls of benign sites.

5.3.1 URL

A URL is an acronym for (Uniform Resource Locator) which is basically a specific type of URI (Universal Resource Identifier) which gives a reference to an existing resource on the Internet. A URL basically consists of several components. To learn the structure and the components of the url , we will use the following example of the URL:

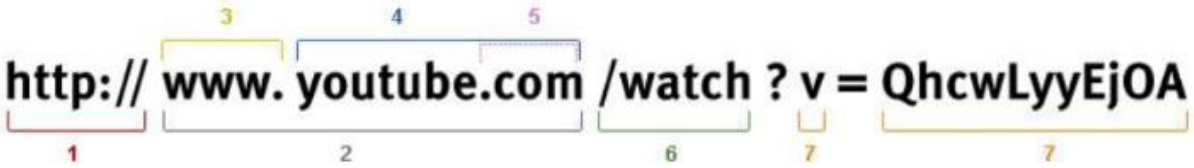


Fig 5.3.1 URL

1. The Protocol in use in this case: HTTP (Hypertext Transfer Protocol).

   There are also other protocols like HTTPS, FTP, MAILTO and so on. It refers to the name of the protocol to be used to obtain the resource.

2. The Host or Hostname: www.youtube.com.

   It refers to the name of the web server on which the resource is available. It is basically, the "domain" to which the URL is referring.

3. The Subdomain: www.

   It is a domain that is a part of a main domain.

4. The domain name (Domain): youtube.com .

   IP addresses are determined using Domain names.

5. The Top-Level-Domain (a web-address suffix): .com.

   Also known by the shorthand TLD it refers to the last segment of the domain name.

6. The Path: '/watch'.

   A path usually points out to a file or folder (directory) on the machine (for example "/folder/file.html").

7. Parameter and value: v (Parameter), QhcwLyyEjOA (Parameter value).

   Parameters are initialised by the "?" inside the URL. In the given url, "v" is the parameter name and the value of the parameter is "QhcwLyyEjOA" (Name of the parameter and its value always have the same structure: Parametername = Parametervalue).

5.3.2 Data Collection

The URLs of benign websites were collected from (The Web information company – www.alexa.com) and phishing websites from (Phishtank: www.phishtank.com, 2017).

Also Url's are manually collected and some are collected from the various trusted github files. The dataset consists of large number of urls which are then divided into Training and Testing Dataset usually in ratio of 80:20 or 90:10. This also helps us in finding the efficiency of our classifier model built. Here, our Url dataset contains only domain names and other field as malicious i.e. binary in value (0 or 1) representing as malicious or benign.

	url	isMalicious
2	diaryofagameaddict.com	1
3	espdesign.com.au	1
4	iamagameaddict.com	1
5	kalantzis.net	1
6	slightlyoffcenter.net	1
7	toddscarwash.com	1
8	tubemoviez.com	1
9	ipl.hk	1
387584	youthleaguesusa.com/potomacsoccer/11-12/Tryout.html	0
387585	youthleaguesusa.com/potomacsoccer/2011/Tournament.html	0
387586	zip-codes.com/	0
387587	owens.edu/news-releases/?p=2052	0
387588	1.safesecureweb.com/egale/index.asp?item=1173	0
387589	yurika.otakuthon.com/reg/main.pl/en/	0
387590	hottraveljobs.com/forum/docs/info.php	1
387591	news.grouptumbler.com/news/feed.php	1
387592	info.leveldelta.com/php/text.php	1
387593	citroen-club.ch/n.exe	1
387594	zehir4.asp	1
387595	ZHC_Shell_1.0.aspx	1
387596	img851/2304/bismillahus.jpg	1
387597	himselp.net.in/css/acrord.exe	1
387598	www.skyslisten.com/help.html	1

Fig 5.3.2 Url Dataset

5.3.3 Data Preprocessing and Cleaning

This presents a technique of data mining which transforms extracted raw data into meaningful data. Raw data is mostly inconsistent, incomplete, may lack in certain behavior's and trends and may contain many errors. This process helps to resolve these type of issues. It basically prepares raw data for further processing and data transformation for efficient and effective processing for the purpose of the user.

From the figure of Feature Importance we can clearly that the highest importance is given to

suspicious words and also the maximum length of the url. Then the others features such as count of letter “w”, total dots in the url, number of delimiters in the url, etc. Now as we will be using the word embedding and Lstm model, it depicts the detection using words build or domain name. So during training and predicting we will use the feature maximum length of the url as one of the processing to be done.

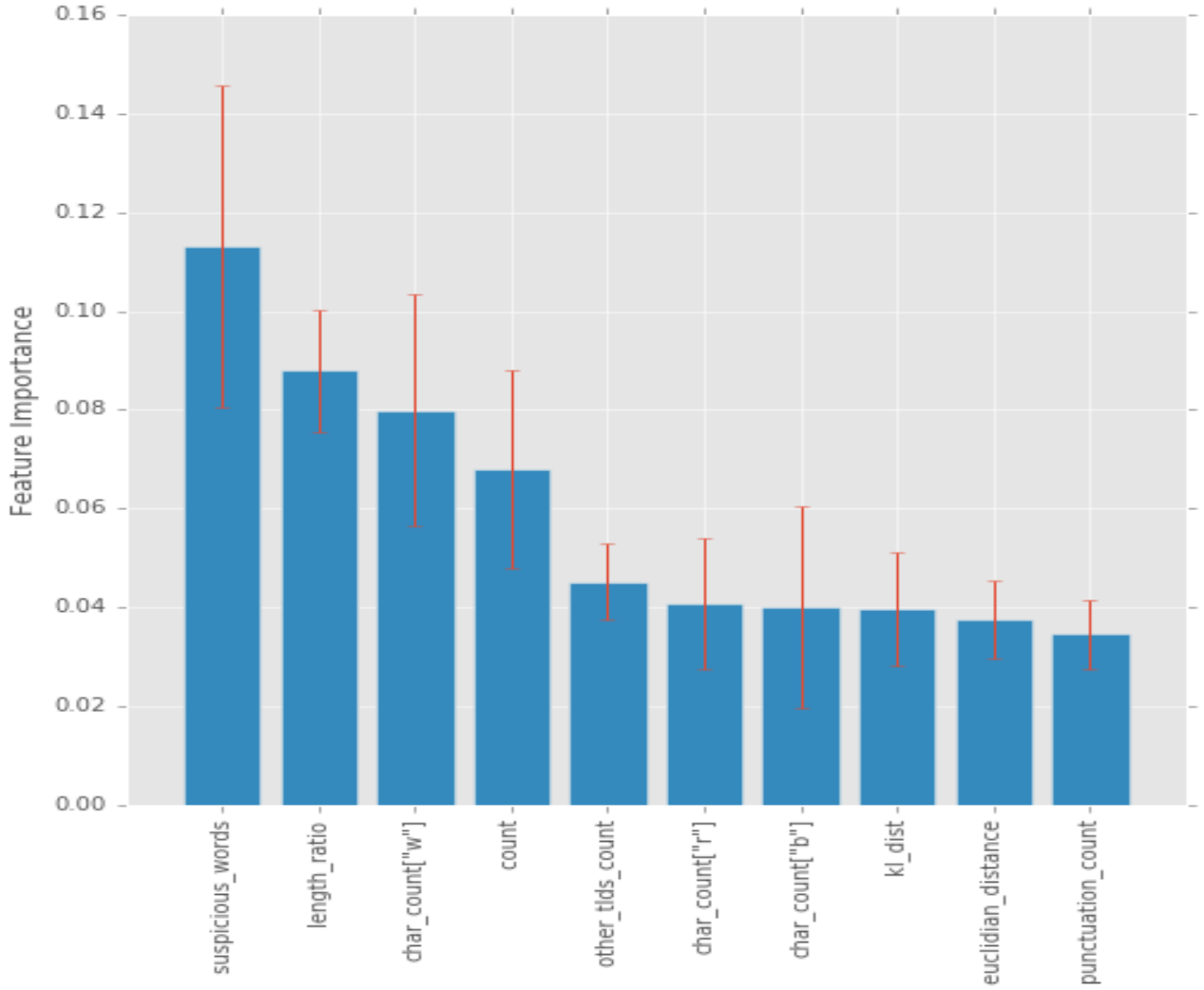


Fig 5.3.3 Feature Importance Plot

5.3.4 Data Integration

In this Method Data is combined from different online sources into a coherent store. Integrating metadata from different sources under schema integration. The most important role of data integration is to remove redundancy and duplicacy in data.

5.4 LSTM Layer

LSTM network is a kind of Recurrent Neural Network (RNN). A Recurrent Neural Network is a neural network that attempts to model time or sequence dependent behaviour – such as language, stock prices, electricity demand, predicting, NLP and so on.

In the below diagram, a chunk of neural network, "A", looks at some input "xt" and outputs a value "ht". A loop allows information to be passed from one step of the network to the next. Recurrent neural networks are "unrolled" programmatically during training and prediction.

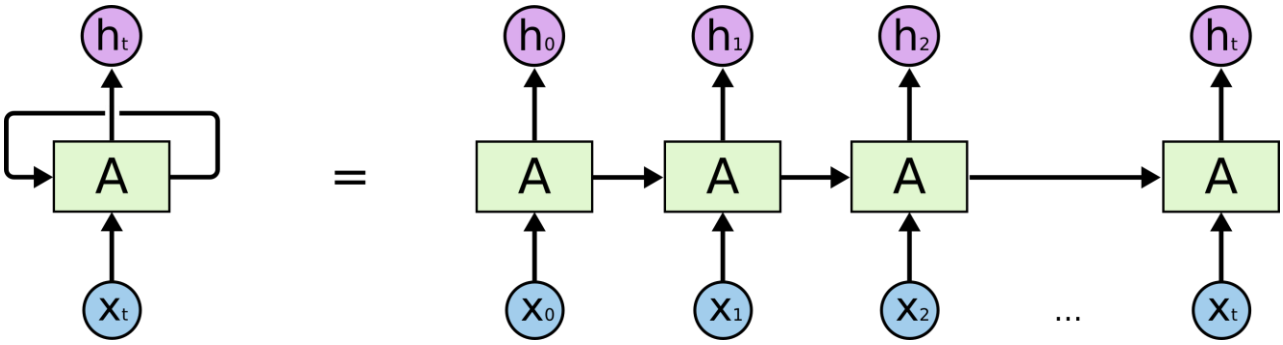


Fig 5.4 Unrolled Recurrent Neural Network

RNNs are absolutely not capable of handling "long-term dependencies". These problem can be easily solved by using LSTM networks.

An LSTM network is a recurrent neural network that has LSTM cell blocks in place of our standard neural network layers. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. These cells have various components called the input gate, the forget gate and the output gate.

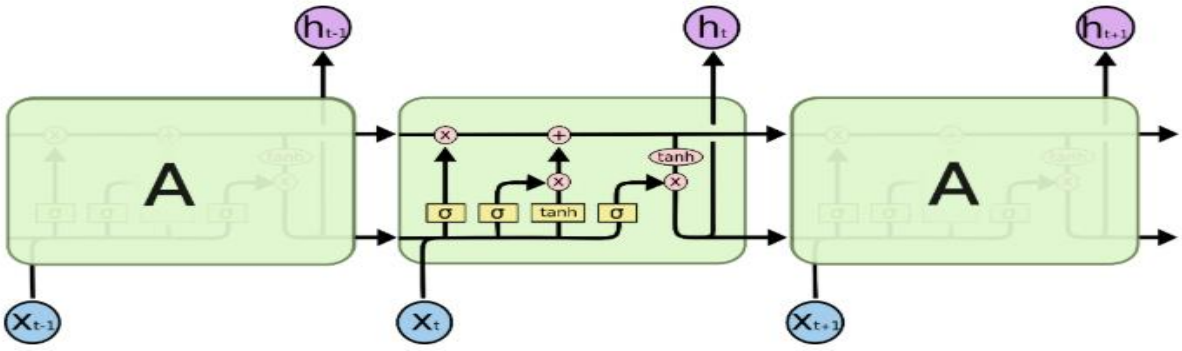


Fig 5.4.1 Recurrent LSTM Cell

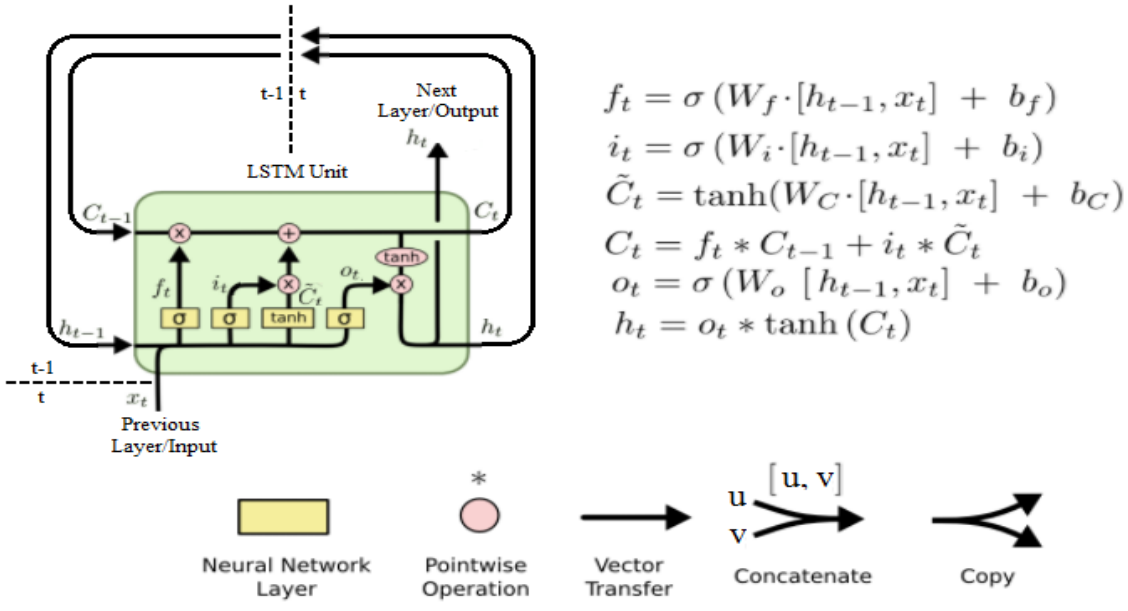
i) The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer (ft)" It looks at ht−1 and xt, and outputs a number between 0 and 1 for each number in the cell state "Ct−1". A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

ii) The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a "tanh" layer creates a vector of new candidate values, "~Ct", that could be added to the state. In the next step, we'll combine these two to create an update to the state.

iii) It's now time to update the old cell state, "Ct−1", into the new cell state "Ct". The previous steps already decided what to do, we just need to actually do it.

We multiply the old state by "ft", forgetting the things we decided to forget earlier. Then we add "it∗~Ct". This is the new candidate values, scaled by how much we decided to update each state value.

v) Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through "tanh" (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fig 5.4.2 LSTM Unit Cell

Chapter 6

Implementation Details

6.1 Training in Google Colaboratory

6.1.1 Google Colaboratory

Google Colaboratory is based on the Jupyter notebook design and operation paradigm. You don't need to install any libraries such as Tensorflow, Keras, etc separately for colaboratory. To access the environment, you must have a Google Drive account and be signed in. The .ipynb files that you create will be saved in your Google Drive account.

To access Google Colaboratory, go here. Once you open up a new file, the first thing to do is rename the file (File -> Rename) and setup your running environment (i.e. whether to use a standard CPU, GPU or TPU). Whenever you change your running environment, the current notebook session will restart – so it is best to do this first up. To do so, go to Runtime -> Change runtime type.

One of the most important and useful components of Google Colaboratory is the ability to share your notebooks with others, and also allow others to comment on your work.

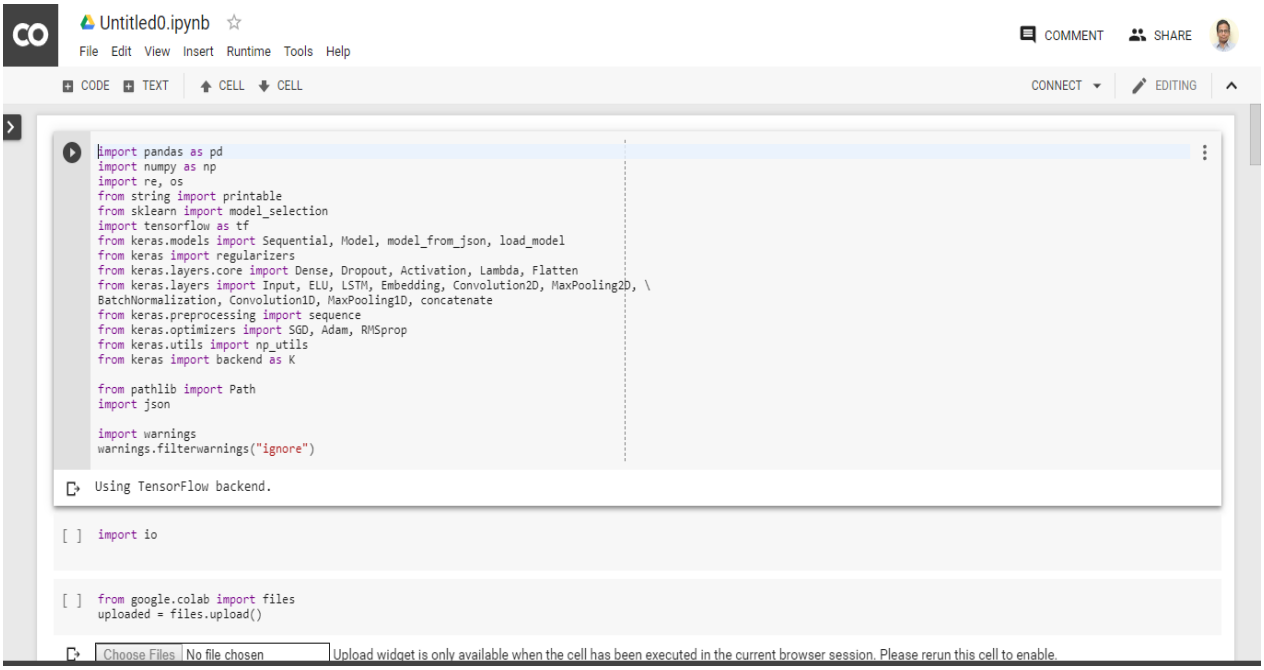


Fig 6.1.1 Google Colaboratory

6.1.2 Data Collection and Processing

The dataset (containing both malicious and benign URLs to train the Deep Learning binary classifier) was custom build from various open source data sources. Some of the open source URLs came with the zone apex only, others didn't include the protocol, therefore, we uniformly removed the protocol (http:// or https://) and the subdomain (e.g. www) from the URL string if applicable.

For the Featureless Deep Learning approach, no manual features are needed. However, limited pre-processing of the raw URLs is still necessary. The raw URL string needs to be split into "words". Very easily done, every single character can be considered a "word". In addition, each character has to be expressed as unique integer. This requires building a dictionary first. A short cut can be considering Python's 100 printable characters only.

Fig 6.1.2 URL to Integer

All URLs have to be of the same length. This results in cropping or padding with zeros. I choose a max length of 75 characters.

max_len=75

X = sequence.pad_sequences (url_int_tokens, maxlen=max_len).

6.1.3 Model Preparation

The featureless part is accomplished through word2vec. The neural network is trained first to "embed" characters that occur in the same "context", that is, are close-by in a n-dimensional space (You can tune and randomly pick). After that model has been trained the "real" deep neural network is trained.

Here we define the vocabulary size as 100 (since we used Python's 100 printable characters only) and embedding dimension as 32 (as per tuning). Each character contained in the vocabulary has its own embedded vector. Below is a visualization of the word2vec embedding model matrix by projecting the n-dimensional vectors on 2D (e.g. using t-SNE). Interesting relationships of characters that tend to appear in similar contexts can be seen.

The word2vec is trained first, then applied to each URL to "embed"/transform the URL and thus derive it's "features", so that the actual binary classifier can be trained thereafter. Every single embedded URL is of shape:

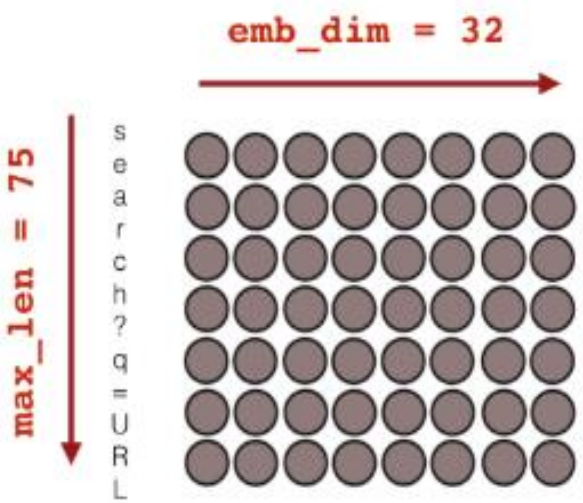


Fig 6.1.3 Shape of Embedded Url

Here, we build the model as follows:

The very first initial layer is always an input layer where you define the initial input shape (here initial 75 characters of the URL)

a) main_input = Input(shape=(max_len,), dtype='int32', name='main_input')

The next line adds an embedding layer.

b) emb = Embedding(input_dim=max_vocab_len, output_dim=emb_dim, input_length=max_len, dropout=0.2, W_regularizer=W_reg)(main_input)

The next line adds an LSTM layer. This is the main part of our technique. 32 represents the number of dimensions in our internal state.

The Dropout layer is a trick used in deep learning to prevent overtraining. You can probably remove this, but we found it useful.

c) lstm = LSTM(lstm_output_size)(emb)

d) lstm = Dropout(0.5)(lstm)

This Dropout layer precedes a Dense layer (fully connected layer) of size 1.

We added a sigmoid activation function to squash the output of this layer between 0 and 1, which represents, benign and malicious.

e) output = Dense(1, activation='sigmoid', name='output')(lstm)

And finally we compile the model using adam optimizer or other to calculate the efficiency of the model.

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

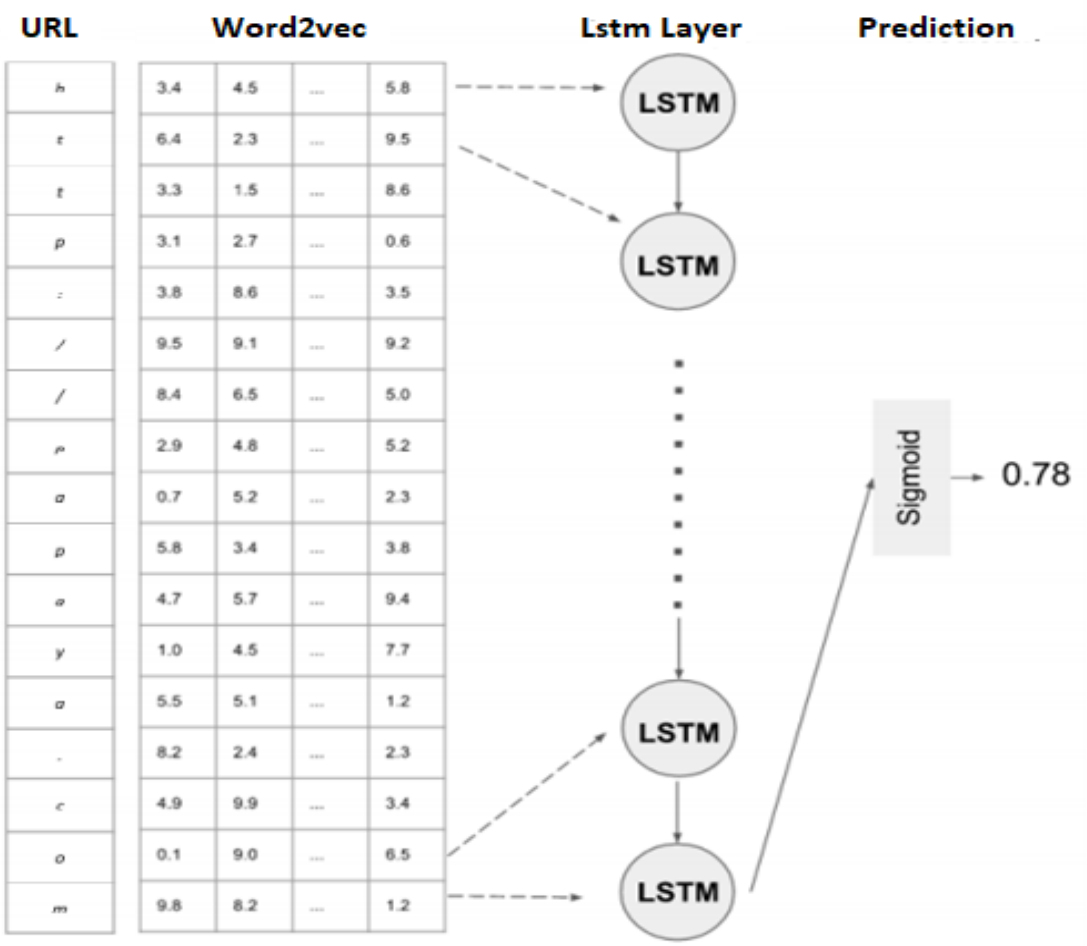


Fig 6.1.4 LSTM Model

6.2 GUI Module

We are basically using Tkinter for developing our GUI. Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, Tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with Tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using Tkinter is an easy task.

To use a Tkinter:

1. Importing the module – Tkinter.
2. Create the main window (container).
3. Add any number of widgets to the main window.
4. Apply the event Trigger on the widgets.

Importing Tkinter is same as importing any other module in the python code. There are two main methods used, which the user need to remember while creating the Python application with GUI.

1. Tk (screenName=None, baseName=None, className=Tk, useTk=1): To create a main window. To change the name of the window, you can change the className to the desired one.
2. mainloop(): There is a method known by the name mainloop() is used when you are ready for the application to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

Tkinter provides the following widgets:

i) Button:- Button widget is used to place the buttons in the tkinter.

ii) Canvas:- Canvas is used to draw shapes in your GUI.

iii) Checkbutton:- Checkbutton is used to create the check buttons in your application. You can select more than one option at a time.

iv) Entry:- Entry widget is used to create input fields in the GUI.

v) Frame:- Frame is used as containers in the tkinter.

vi) Label:- Label is used to create a single line widgets like text, images, etc.

vii) Menu:- Menu is used to create menus in the GUI.

The sample Gui using Tkinter Module is given here:

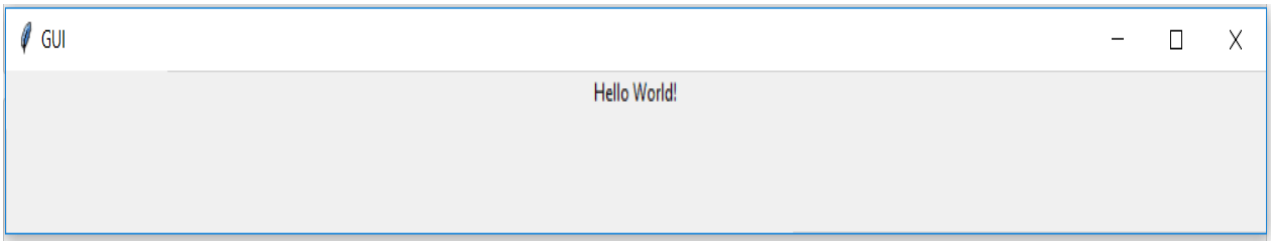


Fig 6.2.1 GUI Sample

Our GUI has a text box in which user can enter the url for which he/she has to check the phishing status. On clicking the submit button on the gui ..it will give the result as:

1. The URL (www.something.com) is Malicious(if it is phishing website)

2. The URL (www.something.com) is Benign(if it is genuine website)

Fig 6.2.2 System GUI

The url (https://www.facebook.com) is written in the text box .As this is the genuine website as we know so the result shown here is : "The URL (https://www.facebook.com) is Benign".

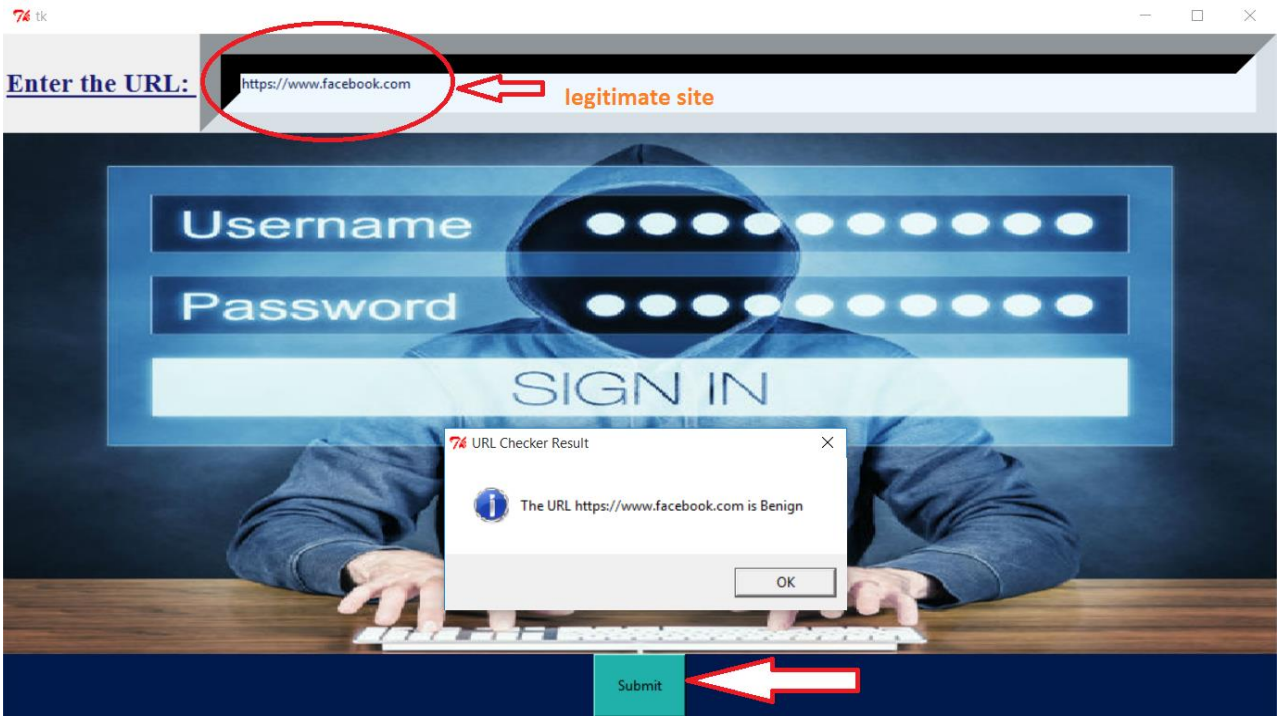


Fig 6.2.3 Smart Malicious Url Detection System

The url (http://www.facebook.pcriot.com/login.php) is written in the text box .As this is the phishing website as we know so the result shown here is :

"The URL (http://www.facebook.pcriot.com/login.php) is Malicious".

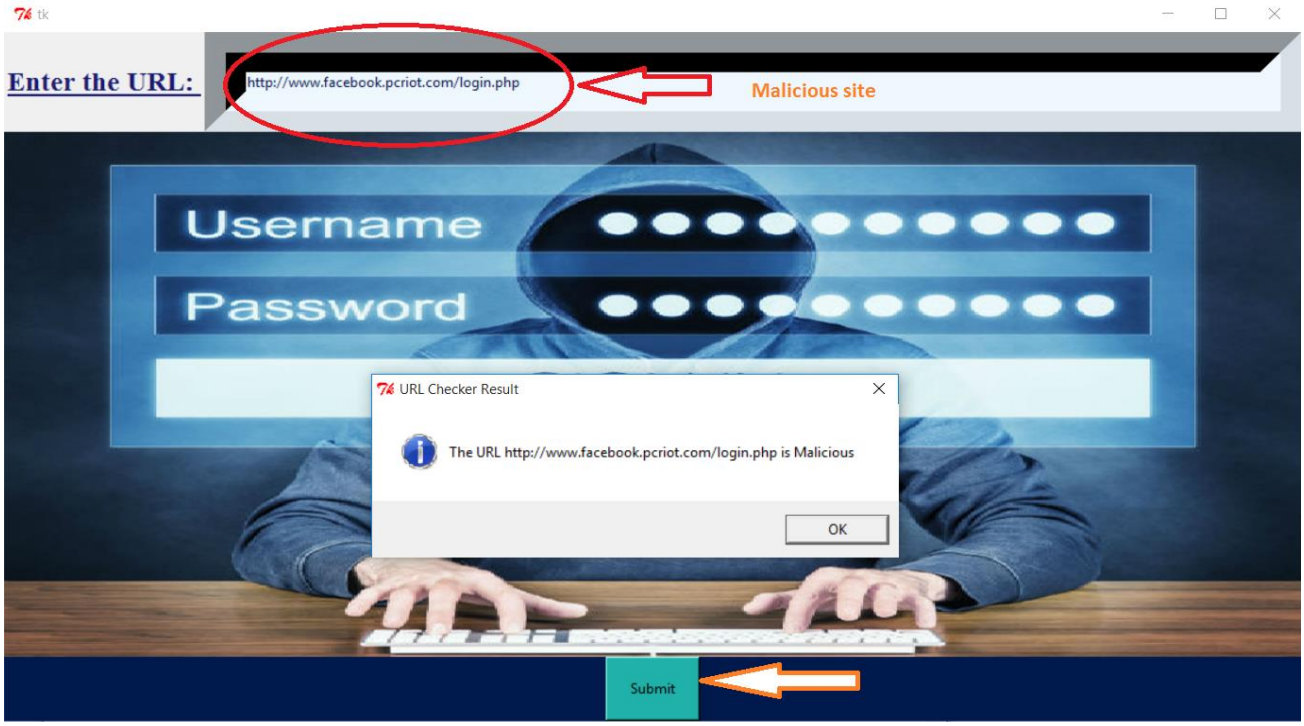


Fig 6.2.4 Smart Malicious Url Detection System

6.3 Google Chrome Extension

Extensions are small software programs that customize the browsing experience. They enable users to tailor Chrome functionality and behavior to individual needs or preferences. They are built on web technologies such as HTML, JavaScript, and CSS. User interfaces should be minimal and have intent. Extension files are zipped into a single .crx package that the user downloads and installs. This means extensions do not depend on content from the web, unlike ordinary web apps. Extensions are distributed through the Chrome Developer Dashboard and published to the Chrome Web Store.

Installing Created Google Chrome Extension:

i) Navigate to chrome://extensions in your browser.

ii) Check the box next to Developer Mode.

iii) Click Load Unpacked Extension and select the directory for your "Smart Url Detection System" extension.

As Extension is easy to use and provides quicker accessibility, we created a extension for system using Html, Css and Javascript. Now we need a web page to click the button to start the detection of url. So we host our website currently on Xampp server. Our extension automatically takes the url from the search bar for the detection, thus minimizing the effort of the user. The extension can be easily on the Right hand side top corner of the google chrome with a small icon.

Fig 6.3.1 Smart Malicious Url Detection System Extension

When the user clicks on the SAFE or Not? Button or hovers on it, button turns from green to white and it takes the url from the search bar and runs it to the classifier model.

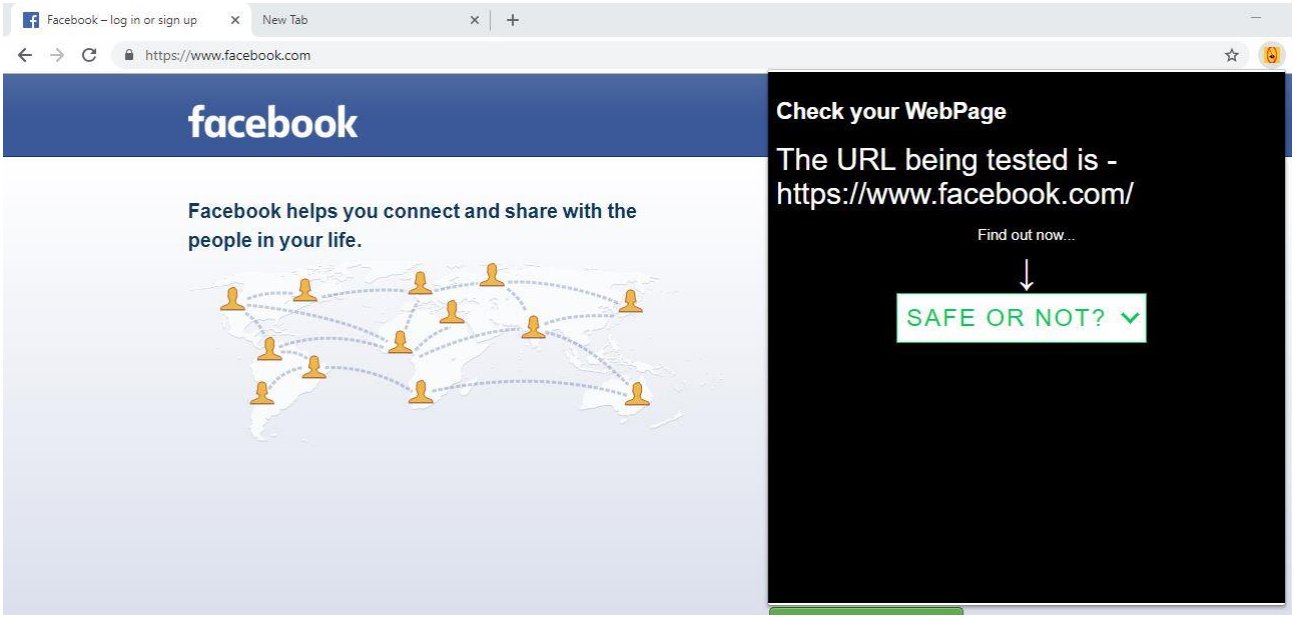


Fig 6.3.2 Extension Page

The Result is then displayed to the user, when it clicks the SAFE or NOT? Button. Here we can it for https://www.facebook.com

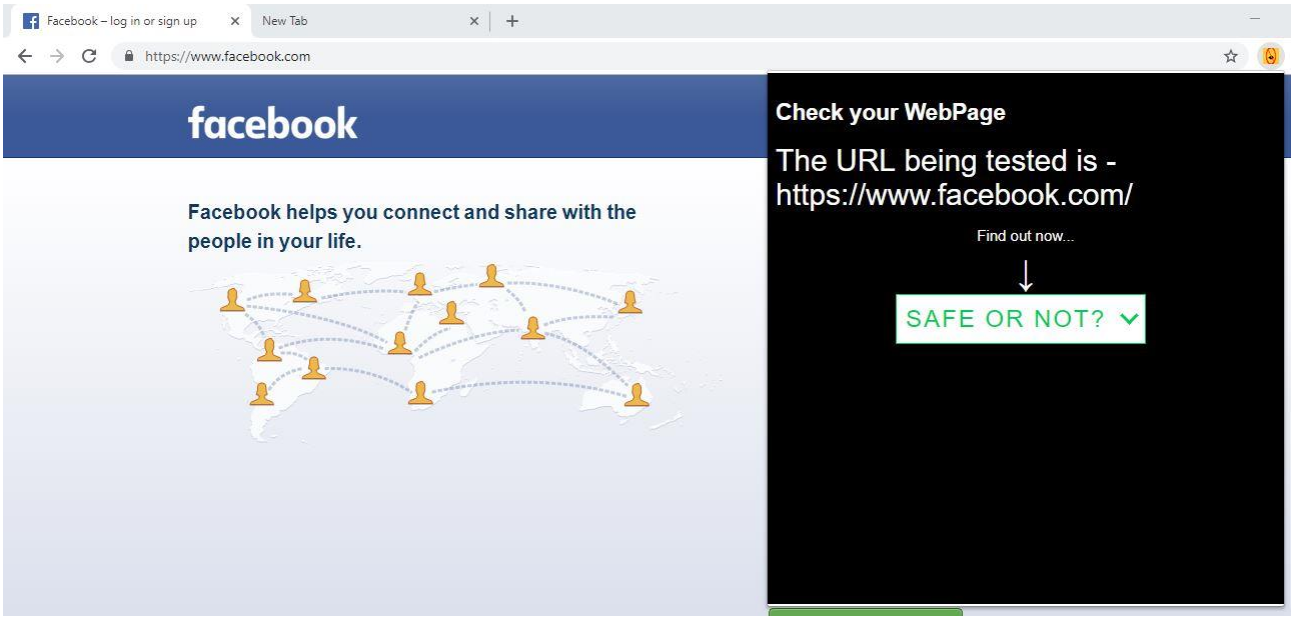


Fig 6.3.3 SAFE Website Example

Thus, the Result is displayed as “SAFE”.

Now if the website is https://facebook.com-mobi-qpmytzupfr.montconghana.com/aba

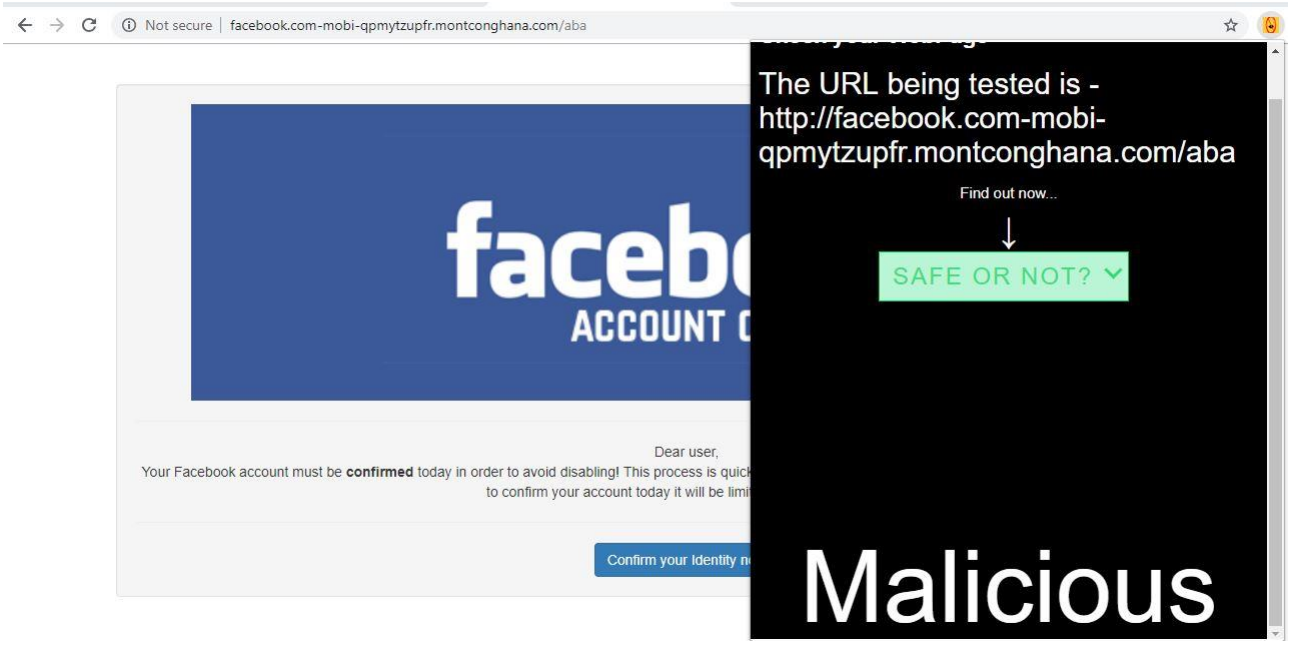


Fig 6.3.4 Malicious Website Example

Thus, the Result is displayed as “Malicious”.

Chapter 7

Testing

7.1 Test Cases

The table below specifies different test cases for the Smart Malicious Url Detection :-

Test Case Id	Input	Expected Output	Actual Output	Result
T001	https://www.shahandanchor.com	Benign	Benign	Pass
T002	Https://www.facebook.com	Benign	Benign	Pass
T003	https://www.facebook.pcriotcom/login.php	Malicious	Malicious	Pass
T004	https://www.dslreports.com/forum/r32332858-Microsoft-Security-Update-MinorRevisions-Issued-March-21-2019	Benign	Benign	Pass
T005	http://plusdomeuescritorio.com.br/sicronizacaodedadosseguroprotocolo/br/essoa-juridica/home.php	Malicious	Malicious	Pass
T006	https://kingroot.en.uptodown.com/android	Benign	Malicious	Fail
T007	https://app-19695722944.000webhostapp.com/	Malicious	Benign	Fail
T008	https://lfood-bonus-com.umbler.net/bancointer.com.br/oflogin/	Malicious	Malicious	Pass
T009	https://www.idbi.com/idbi-bank-internet-banking.asp	Benign	Benign	Pass
T010	https://www.youtube.com/channel/UCOhHO2ICt0ti9KAh-QHvttQ	Benign	Malicious	Fail

Table 7.1 Test Cases

Chapter 8

Result & Analysis

Phishing is considered as the most easiest way to attack user and steal it's data. To avoid this many Url detection systems are built using blacklist and whitelist data, using feature extraction and training them on traditional machine learning algorithm. But both of them are not good enough as first method identifies malicious urls which are only blacklisted and entered earlier in the list whereas second method extracts specific lexical and host based features and are classified using KNN, SVM, linear regression and so on, but all these classifiers accuracy is around 85%.

Responses to user requests are delivered to the users using Graphical interface. User needs to enter the URL in the text box of GUI and then user will get to know the phishing status of the URL he entered. We are exploring the LSTM algorithm in neural networks which is providing an accuracy of about 95% and removes manual feature selection process by using RNN i.e specifically Long Short Term Memory (LSTM) by embedding Url using word2vec and creating a feature set for it. This approach uses a character sequence pattern to create the feature set. We have created an Google chrome extension for this system in which the URLS will be extracted using Javascript and then they will be executed for the result by using python code i.e. using LSTM classifier model.

Chapter 9

Conclusion & Future Scope

Insufficient knowledge and consciousness on phishing education makes such malicious attacks successful. Even the few indicators used by the browser such as pad lock identification, lock icon, and site identify button, don't help and the user still cannot identify the attack. Additionally, the users should not follow links to sites blindly where they need to enter the delicate information. It is vital to check the URL before entering the site.

The principal motivation of this study was to assist the users in analysing the legitimate web page and fake web page by using URL as an indicator. We proposed a Smart Malicious URL phishing detection system for end user with the following characteristics:

- The GUI of our system will engage end users and provide user friendly experience with a (.exe) extension support.
- LSTM Model with an accuracy of around 95%.
- System will be based on discerning Urls by their patterns.
- No manual Feature Extraction.

Various other modules can be added to it regarding different types of cybercrime and its information i.e. all information that user should be aware of. Also, one can add other scanning modules such as File scan, Virus scan in the same project and make it a fully complete Security Application. We can try to make the program operating system friendly so it can be runned on of the platforms. We could also make an API for the software so it can be used by other applications and companies or users easily with other programs.

Appendix

A

Artificial Neural Network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output.

Artificial Intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans.

D

A **Deep Neural Network** (DNN) is an Artificial Neural network (ANN) with multiple layers between the input and output layers.

F

Fuzzification is the process of changing a real scalar value into a fuzzy value. This is achieved with the different types of fuzzifiers (membership functions). Fuzzification. Fuzzy Linguistic Variables are used to represent qualities spanning a particular spectrum. Temp: {Freezing, Cool, Warm, Hot}.

K

K Nearest Neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

M

Machine Learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

R

Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step**.**

References

[1] Mohammed Al-Janabi, Ed de Quincey, Peter Andras, “Using supervised machine learning algorithms to detect suspicious URLs in online social networks”. In Advances in Social Networks Analysis and Mining 2017 on (pp. 1104-1111). ACM.

[2] Qiongxia Huang, Xianghan Zheng, Riqing Chen and Zhenxin Dong, “Deep Sentiment Representation Based on CNN and LSTM”. In 2017 International Conference on Green Informatics (ICGI). IEEE.

[3] A.Bavani, D.Aarthi, V.C,2017, Detecting phishing websites on real time using anti-phishing framework, Department of Information Technology (UG), Assistant Professor of Kingston Engineering College, India.

[4] Adulghani Ali Ahmed, N. A. A.: “Real time detection of phishing websites”. In 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) IEEE.

[5] Ankit Kumar Jain, B.B. Gupta : A novel approach to protect against phishing attacks at client side using auto-updated whitelist. In EURASIP Journal on Information Security (2016(1)).

[6] Common Crawl: www.commoncrawl.org.

[7] Phishtank: www.phishtank.com.

[8] Kaggle: www.kaggle.com.

[9] Tensorflow: https://tensorflow.org/tutorials/representation/word2vec.

[10] Keras: https://keras.io/layers/recurrent/

[11] LSTM Deep Learning for NLP: https://medium.com/@shivambansal36/language-modelling-text-generation-using-lstms-deep-learning-for-nlp-ed36b224b275

Acknowledgement

We sincerely express our deep gratitude to our Principal Dr. Bhavesh Patel, our Head of Department Mr. Uday Bhave and our guide Prof. Deepti Nikumbh for providing us with their invaluable guidance, advice and suggestions. We were fortunate to have met such supervisors. We would like to thank our guides for providing us with the opportunity to do this project Smart Malicious Url's Detection System to Prevent Phishing Using Deep Learning Approach, which helped us in doing lot of research and learning new things. We acknowledge with a deep sense of gratitude, the encouragement and inspiration received from our faculty members and colleagues.