Visvesvaraya Technological University

BELAGAVI, KARNATAKA.

ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ

ಬೆಳಗಾವಿ, ಕರ್ನಾಟಕ

A PROJECT PHASE-2 REPORT

ON

"Anti-Phishing : A Web Identifier For Spoofed Sites Using Neural Network "

Submitted to Visvesvaraya Technological University in partial fulfillment of the requirement for the award of Bachelor of Engineering degree in Computer Science and Engineering.

Submitted by

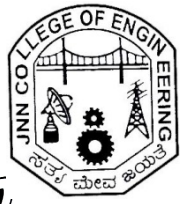
Name	USN
Akshatha A P	**4JN20CS005**
Chaithra R	**4JN20CS024**
Madhura S	**4JN20CS050**
Chandana C Sagar	**4JN20CS127**

Under the guidance of

Mr. Hiriyanna G S B.E.,M.Tech.

Asst.Professor, Dept. of CS&E,

JNNCE, Shivamogga.



Department of Computer Science & Engineering

Jawaharlal Nehru New College of Engineering

Shivamogga - 577 204

May 2024

National Education Society ®

Jawaharlal Nehru New College of Engineering

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled

"Anti-Phishing : A Web Identifier For Spoofed Sites Using Neural Network "

Submitted by

Name	USN
Akshatha A P	**4JN20CS005**
Chaithra R	**4JN20CS024**
Madhura S	**4JN20CS050**
Chandana C Sagar	**4JN20CS127**

Students of 8th semester B.E. CS&E, in partial fulfillment of the requirement for the award of degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the year 2023-24.

Signature of Guide

Mr. Hiriyanna G S B.E.,M.Tech.,
Asst.Professor, Dept. of CS&E,

Signature of HOD

Dr. Jalesh Kumar B.E., M.Tech., Ph.D.,
Professor and Head, Dept. of CS&E

Signature of Principal

Dr. Y Vijaya Kumar B.E., M.Tech., Ph.D.,
Principal, JNNCE

Examiners: 1._____ 2. _____

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Akshatha A P

in recognization to the publication of paper titled

Literature Review on Phishing Website Detection Using Deep Learning

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Chaithra R

in recognization to the publication of paper titled

Literature Review on Phishing Website Detection Using Deep Learning

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Madhura S

in recognization to the publication of paper titled

Literature Review on Phishing Website Detection Using Deep Learning

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Chandana C Sagar

in recognization to the publication of paper titled

Literature Review on Phishing Website Detection Using Deep Learning

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM34558

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Akshatha A P

in recognization to the publication of paper titled

Anti-Phishing: A Web Identifier for Spoofed Sites Using Neural Network

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

Editor-in-Chief
IJSREM Journal

www.ijsrem.com e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM34558

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Chaithra R

in recognization to the publication of paper titled

Anti-Phishing: A Web Identifier for Spoofed Sites Using Neural Network

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

Editor-in-Chief
IJSREM Journal

www.ijsrem.com e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM34558

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Madhura S

in recognization to the publication of paper titled

Anti-Phishing: A Web Identifier for Spoofed Sites Using Neural Network

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

Editor-in-Chief
IJSREM Journal

www.ijsrem.com e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM34558

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Chandana C Sagar

in recognization to the publication of paper titled

Anti-Phishing: A Web Identifier for Spoofed Sites Using Neural Network

published in IJSREM Journal on Volume 08 Issue 05 May, 2024

Editor-in-Chief
IJSREM Journal

www.ijsrem.com e-mail: editor@ijsrem.com

ABSTRACT

Phishing attacks persist as a significant threat to cybersecurity, exploiting deceptive websites to illicitly acquire sensitive user information. Conventional anti-phishing techniques often struggle to keep pace with the evolving sophistication of these attacks. The primary objective is to train a deep neural network on a carefully curated dataset, distinguishing between legitimate and phishing websites based on their URLs. The project involves implementing a feature-based extraction approach for URLs, utilizing techniques to identify distinctive patterns indicative of phishing attempts. By extracting relevant features directly from URLs and passing it to a deep learning LSTM and Dense training model to learn and recognize the characteristics of both legitimate and phishing websites. The system aims to enhance efficiency and accuracy in distinguishing between legitimate and malicious websites. Leveraging the power of deep neural networks, the model undergoes extensive training to effectively differentiate between the two categories.

ACKNOWLEDGEMENT

We would like to acknowledge our profound gratitude to all those who have helped in implementing this project.

We are grateful to our institution Jawaharlal Nehru New College of Engineering and Department of Computer Science Engineering for imparting us the knowledge with which we can do our best.

We would like to thank our beloved guide **Mr. Hiriyanna G S,** Assistant Professor, Dept. of CS&E who have helped us a lot in making this project and for their continuous encouragement and guidance throughout the project work.

We would like to thank our respected project co-ordinators **Dr. K.M Poornima,** Professor**, Dr. Ganavi M,** Assistant Professor**, Mrs. Pushpa R N,** Assistant Professor, who have extended their warm support with respect to all aspects of project.

We would like to thank **Dr. Jalesh Kumar,** HOD of CS&E Dept and **Dr. Y Vijaya Kumar,** Principal, JNNCE, Shimoga for all their support and encouragement.

Finally, we also would like to thank the whole teaching and nonteaching staff of Computer Science and Engineering.

Thanking you all,

Project Associates,

Akshatha A P	**4JN20CS005**
Chaithra R	**4JN20CS024**
Madhura S	**4JN20CS050**
Chandana C Sagar	**4JN20CS127**

TABLE OF CONTENTS

ABSTRACT		i
ACKNOWLEDGEMENT		ii
TABLE OF CONTENTS		iii-iv
LIST OF FIGURES		v
CHAPTER 1	**Introduction**	**1-17**
	1.1 Literature Survey	2-15
	1.2 Problem Statement	16
	1.3 Objectives	16-17
	1.4 Scope of the Project	17
	1.5 Organization of the Report	17
CHAPTER 2	**Deep Learning Algorithms**	**18-26**
	2.1 Deep Learning	18-19
	2.1.1 LSTM	19-21
	2.1.2 Dense Layer	22-24
	2.2 Selenium	24-26
	2.3 Summary	26
CHAPTER 3	**System Design and its Implementation**	**27-33**
	3.1 System Framework	27-29
	3.2 Implementation	30-33
	3.3 Summary	33
CHAPTER 4	**Result and Snapshots**	**34-36**
	4.1 Result analysis	34-36
	4.2 Summary	36

CHAPTER 5	**Conclusion and future scope**	**37**
	5.1 Conclusion	37
	5.2 Future scope	37
	Publication	38
	References	**39-40**

LIST OF FIGURES

Fig. no.	Fig. name	Page no.
2.1	Unit architecture of LSTM	20
2.2	Dense Layers	22
2.3	Selenium Architecture	24
3.1	System framework	27
4.1	Snapshot of Chrome handled by testing software	34
4.2	Snapshot of Genuine Website	35
4.3	Snapshot of Phished Website	35
4.4	Snapshot of Popup message	36
4.5	Snapshot of Feature analysis	36

CHAPTER 1

INTRODUCTION

Phishing is a form of cyber attack where malicious actors attempt to trick individuals into disclosing sensitive information, such as login credentials, financial details, or personal data. The term "phishing" is a play on the word "fishing," as it involves luring victims with bait. These attacks typically take the form of deceptive emails, text messages, or websites that appear to be from legitimate sources, such as banks, social media platforms, or government agencies.

Phishing attacks often exploit psychological tactics, such as urgency or fear, to prompt victims to act quickly without questioning the authenticity of the communication. For instance, a phishing link might mimic a bank's website URL but feature a subtle alteration, redirecting unsuspecting users to a fraudulent site designed to harvest their login credentials. Unwary users who fall for these tricks may inadvertently provide their sensitive information to cybercriminals.

There are several common types of phishing attacks, including:

1. Email Phishing: This is the most prevalent form of phishing, where attackers send deceptive emails purporting to be from trusted organizations. These emails often contain links to fake websites or malicious attachments designed to steal information.

2. Spear Phishing: In spear phishing attacks, cybercriminals target specific individuals or organizations, often using personalized information to increase the likelihood of success. This could involve using the victim's name, job title, or other details obtained through research.

3. Whaling: Whaling attacks target high-profile individuals, such as executives or celebrities, with the aim of stealing sensitive information or perpetrating financial fraud. These attacks often involve sophisticated tactics and social engineering techniques.

4. Smishing: Smishing, or SMS phishing, involves sending fraudulent text messages to trick recipients into revealing personal information or clicking on malicious links. These messages often claim to be from banks, delivery services, or other trusted organizations.

5. Vishing: Vishing, or voice phishing, involves using phone calls to deceive victims into providing sensitive information or performing certain actions. Attackers may impersonate legitimate organizations or individuals to gain the victim's trust.

Phishing websites are fraudulent online platforms designed to imitate legitimate websites, often with the aim of deceiving users into divulging sensitive information such as passwords or financial details. These malicious sites typically employ various tactics to appear authentic, including mimicking the visual design and branding of reputable organizations or using deceptive URLs. Once users enter their information, cybercriminals exploit it for nefarious purposes, such as identity theft, financial fraud, or unauthorized access to accounts.

With the rapid development of machine learning, there are more and more applications in the field of cybersecurity, and we have proposed a deep learning-based framework to detect phishing links in a real-time web browsing environment. When the URL of the current tab of the browser is predicted to be a phishing link, the current page will receive an obvious warning prompt. The prediction result is obtained by the core prediction service calling a trained model.

1.1 Literature Survey

OrunsoluAbioduna, Sodiya A. Sb, Kareem S.O [1] LinkCalculator –an efficient link-based phishing detection tool. The proposed LinkCalculator anti-phishing scheme is based on an algorithm designed to extract link characteristics from loading URLs to determine their legitimacy. Unlike the other link-based extraction approaches, the proposed approach introduced the concept of the weighting of the incoming request for its prediction without using the machine learning approach. The weighting concept allows the system to prevent superfluous computations on non-essential links within the parsed page. The advantage of this is to reduce the problems of false-positive and negatives occasioned by other methods where this idea is missing. This is because certain link information within parsed webpages or requests is sufficient to classify them as phishing without loss of generality. To this end, the proposed Link Calculator makes the following contributions to anti-phishing study The proposed system involves a phase where a user's request is retrieved from a client's browser that is attempting to connect to any resources on a particular webserver. A request, in this case, may be defined as any transaction requiring connection to a particular web server on the Internet. Usually, such request may be represented as a URL of a webpage or a link within any webpage. When a user opens any request in a browser, a module called Service Handler is created using JSoup Java

implementation to access a Document Object Model (DOM) tree of the downloaded request from a web browser. The Document Object Model is a World Wide Web consortium standard, that offers a programming interface for web documents. That is, the DOM represents the web documents as nodes and object (i.e. object-oriented platform for web pages) which allows programs and scripts to dynamically connect and update the content, structure and style of documents. The advantage of using the DOM is that it was designed to be independent of any particular programming language and thereby making the structural representation of the web document available from a single consistent API. Upon the creation of DOM, the request can further be tokenized/parsed by JSoup. Parse (request) method and getElementById (request) method to extract any useful information e.g. hyperlinks, URL symbols, URL characters etc. in the body of the transaction or webpage. Tokenization is the process through which web document istransformed into a sequence of string characters while parsing is the method of identifying the structure of a document and extraction of data. The parsing and tokenization process helps to further identify the kind of structure within the request e.g. presence of login tags, null links, anchor tag etc. Forinstance, abnormal status detection can be found on a loading URL by specifying the error code in jsoup implementation. Algorithm 1 presents the basic function of the Service Handler (SH) in the client/user interface layer. The following types of links are considered as phishing "fingerprints" in this approach if they are present in a parsed page in a particular specified threshold. A parsed page that does not contain links. Parsed page with an extraordinary number of null links defined on prominent tags. Parsed page with a high ratio of links directing to the foreign domain than own domain. The system is implemented using JAVA programming language and its associate libraries such as JSoup HTML Parser (JHP), Secure Socket Extension (SSE), etc. These libraries facilitate efficient link extraction from the querying page. These links are then examined by the application for determining the status of the webpage. Other links such as dead links, static links and external links are also defined within the application module to improve page classification. The dead links are links with no action attached to them which are often used by developers to execute scripts and perform actions. On the other hand, the static links are links that lead nowhere. These links, in most cases, are used by malicious websites who have no content to place under these links, so they use static links in making the user believe they have enough data to run their website. External links are links that lead to other websites that are not part of the website that has been loaded by

the user. Scripts such as file_get_content method are employed to retrieve the details of a website. In this process, the scripts download the source code that is readable by the browser. In the process, the Link calculator picks up the webpage source's code from the browser and examines its legitimacy by checking the page conformity with the malicious script definition within the proposed system. To accelerate the process of loading the URLs into the calculator, a Link Loader app was developed to automate the procedure.
The tool introduces a novel algorithm that focuses on extracting link characteristics from loading URLs to assess their legitimacy. The tool has been implemented using the JAVA programming language, utilizing several libraries. This choice of implementation language could offer advantages such as platform independence and a wide range of available libraries for web-related tasks. Unlike other link-based extraction methods, LinkCalculator incorporates the concept of weight into incoming web link information. This is aimed at achieving efficient representation, suggesting a more nuanced understanding of link manipulation. The tool's performance has been evaluated through experiments using standard metrics like False Positives rate and True Positive rate. The plan includes testing the tool on a large dataset using an incremental link analysis method, which could contribute to the tool's scalability and real-world applicability.

Md. Faisal Khana and B. L. Ranab [2] Detection of Phishing Websites Using Deep Learning Techniques.The aim is to detect malicious URLs using minimum features by applying deep machine learning techniques. As an input web-page URLs are fed into the feature extractor. The feature extractor extracts the requisite features from the sources such as from URL, hyperlink and third party based and transfers them to Information Gain (IG) feature ranking algorithm. The IG algorithm supports in choosing the best performance features. The finest performance features are again trained over Deep Neural Network (DNN) to find out the output status and to differentiate between legitimate and phishing URLs.The features have been extracted from URL obfuscation features, hyperlink-based features and third-party based features respectively. These features are extracted using Python with Selenium, a HTML parser and beautiful soup for parsing the webpages. The choice of protruding features from the extracted features is carried out by using IG algorithm.These are the appearances which are extracted directly from URLs, excluding website contents on third party services. Let us discuss the structure of a URL, a URL is a readable text, designed to replace IP addresses that computer systems use to

interconnect with servers. A URL is easily able to determine and analyze a file structure on the given web-page. The attainment of right value ensures that the phishing websites can then be classified with maximum possibility. In our work we have used MATLAB programming to implement deep learning algorithms. After applying a number of hidden layer combinations, the deep neural network (DNN) having five hidden layers gave the best result. The proposed DNN is comprised of 8 layers with 6 unseen layers along with one input. Here, a robust system based on deep learning neural network (DNN) is proposed which is highly efficient in detecting phishing websites. To train the deep learning model, URL heuristics and third party based features have been used. Here we have minimized the number of features as compared to Rao and Pais, thereby reducing the dependence on third party based amenities which is able to attain an accuracy of 99.90%. In future we would like to use more heuristic features which may help in detection of phishing websites faster and more accurately even if the website includes embedded objects.

Seok-junbu, Sung-bae cho [3] proposed Deep Character-Level Anomaly Detection Based on a Convolutional Autoencoder for Zero-Day Phishing URL Detection. In this paper they propose the combination of a convolution operation to model the character-level URL features and a deep convolutional autoencoder (CAE) to consider the nature of zero-day attacks. Extensive experiments on three real-world datasets consisting of 222,541 URLs showed the highest performance among the latest deep-learning methods. they demonstrated the superiority of the proposed method by receiver-operating characteristic (ROC) curve analysis in addition to 10-fold cross-validation and confirmed that the sensitivity improved by 3.98% compared to the latest deep model. They visualize the benign and phishing URL space classified by existing supervised machine-learning techniques using URL observations collected from the PhishTank database. Authors point out, that it was confirmed that keywords such as wp, admin, and content from default settings in the personal server and php keyword can be used as abnormal features of phishing URLs that phishing URLs are particularly longer than benign(safe) URLs and have a composition that is significantly different from the alphabetic distribution constituting natural language. Three main statistics supporting the strong need for character-level modelling in the phishing URL detection task: (a) mutual information by keyword; (b) availability of the URL length feature; (c) character distribution that

separates benign and phishing URLs. The convolution operation aims to learn a spatial filter to extract features in the local receptive field that shares weight, and the long short-term memory (LSTM), a variant of an RNN, is a memory cell that stores the weights used for mapping between inputs and outputs. The learning method is mainly categorized into four approaches: a supervised approach that learns the phishing URL feature and its selection method directly from the label classification result, a semi- or weakly supervised approach that uses only a small number of labels or noisy labels to consider the realistic constraints, and an unsupervised approach that does not use label information of URLs, and an autoencoder-based anomaly detection approach. The fact that phishing URLs are not used to learn the benign URL model in the unsupervised approach is an advantage in class imbalance conditions and, more importantly, is an amenable solution for modelling the nature of a zero-day attack. They used ISCX-URL-2016, Web-accessible Phishstorm and Phishtank datasets are used and collected 95,541 and 60,000 URLs. Class-weight algorithm, t-SNE algorithm are used in this paper.

The main innovation of this study is the introduction of deep anomaly detection to the field of phishing URL detection and achieving the best performance compared to classification-based deep-learning methods by implementing a neural network structure and an operation optimized for URL modelling. The combination of the encoding/decoding structure to facilitate disentanglement between classes and convolution operation optimized for character-level URL characteristics was utilized to define an anomaly score based on the reconstruction error. The limitation of the proposed methodology is that it was optimized for character-level features among the various features constituting URLs. We discussed that the confusion of the character-level features is the main cause of the performance degradation of the proposed method. Considering the structure of the web address consisting of domains and subdomains, additional performance improvements can be expected by utilizing the word-level features, including the typos and the keywords listed in the blacklist.

Lizhen Tang, Qusay H. Mahmoud [4] A Deep Learning-Based Framework for Phishing Website Detection. Phishing is a fraudulent practice in which an attacker masquerades as a reputable entity or person in an email or other form of communication. Attackers use phishing websites to distribute malicious links or attachments that can extract login credentials, account numbers and other personal information from victims. The proposed

approachdepicts the architecture of the components of our proposed framework. There are four modules in terms of data collection tasks, machine learning (ML), cloud application, and web browser extension. The data collection module is an independent scheduled task application. The ML module is used for training modules. The web browser extension is a client-side product. The cloud application is built to deal with false alarms and phishing URLs reported by users from the web browser extension. The orange lines with arrows in the figure show the data interaction process. The core process of this framework is mainly divided into the following six steps: The first is to collect and integrate data from various data sources; which is divided into two parts, obtaining data from different data sources, then analyzing and storing data. The second step is to combine different data sets for machine learning model training, and store the trained model in a file system; {This research developed six machine learning models, namely Logistic Regression, Support vector machines (SVM), Random Forest, RNN, RNN-GRU, and RNN-Long short-term memory (LSTM). Parameter Configuration. Data loading. Feature extraction: natural language processing, doc t matrix 1 token =1 word. In deep learning url to list of character(

ASCII) modelling: RNN -several hidden layer -LSTM(hidden layer) GRU both enhance rnn optimizer and loss function: dump model to file system }.The third step is that the interface for predicting phishing risk calls the trained model to make predictions. The fourth step is that the browser extension calls the prediction interface to perform real-time detection and display the detection results. The fifth step is that users can submit real-time feedback when they disagree with the detection results, such as misjudgement, missed alarm. Finally, the report submitted by the user is verified through manual review and automatic review strategy, and the verification result is synchronized to the data set. Chrome browser plug-in development uses three web front-end development languages: HTML, JavaScript, and CSS. Python and flask for web framework.

The proposed approach has an advantage of Real time browsing environment, feedback data from user enhanced accuracy. It has the capability of running in real time without delays. Also the experimental data can be tracked, real time database, browser extension and blacklist filtering and computer vision services can be added. independent of third party. However the training takes time and to process large number of datasets, it become tedious tasks.

Rundong Yang, KangfengZheng, Bin Wu, Chunhua Wu and Xiujuan Wang [5]Phishing Website Detection Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning. This method uses character embedding, CNN(Convolutional Neural Network) and RF(Random Forest). Firstly URL data transformed into character vector using character embedding that convert URLs to normalized matrices. The model is trained using transformed data using CNN. The features extracted gets classified in random forests. The first to seventh layers are the input, convolutional, pooling, linear 1, linear 2, linear 6, and output layers. Ensemble classification is the classification of phishing websites can be achieved using multi-level features to improve the accuracy and generalization ability of the classification algorithm. URL features are extracted using the pooling layer, L1 layer, and L3 layer. Each RF classification contains 100 decision trees with a maximum depth of 5 in the child nodes.

The proposed approach has the advantage of strong generalization ability, the low-level features in the hidden layer are common and similar for different but related distributed datasets or tasks; these are combined with the low-latitude features in the hidden layer. Third-party service independence, the proposed method relies only on website URL features for detection, without extracting third-party features. Independence of cybersecurity experts, reduced required expert function engineering. Language-independent, the approach proposed in this paper is effective for the detection of websites with content in various languages using character-level features. However the model cannot determine whether the URL is active or not, so it is necessary to test whether the URL is active or not before detection to ensure the effectiveness of detection. Some attackers use URLs that are not imitations of other websites, and such URLs will not be detected. Work aims to use new techniques to automatically extract other features for detecting phishing sites, such as web code features, web text features, and web icon features.

Tristan Bilot, Gregoire Geis and Badis Hammi [6] PhishGNN: A Phishing Website Detection Framework using Graph Neural Networks. Unlike other neural network architectures, GNNs(Graph Neural Networks) can handle non-euclidean data with complex relations between objects. Most GNNs follow the message-passing framework (MPNN) and can be considered as a generalization of convolutional neural networks

(CNN) for graphs. This message-passing algorithm takes as input a graph G = (V,E) with n nodes vi ∈ V and m edges (vi, vj) ∈ E, where G could be directed or undirected.

PhishGNN, a framework for websites classification (phishing or benign) based on Graph Neural Networks. This framework can be considered as an additional layer to GNN architectures. This architecture is divided into two steps namely Pre-Classification and Message Passing. In Pre-Classification, Initially, the graph comprises n nodes (URL), where each node xi($1 \leq i \leq n$) is a vector of d features(URL features) extracted from the corresponding ith URL. x1 is the root URL node(website) and every node xi ($1 < i \leq n$) represent a link coming from x1. At this first step, a binary classifier is used to predict in a semi-supervised mode whether a node is phishing or benign, for each feature node xi ($1 \leq i \leq n$). The classifier is a function g: R d → B, where B is the Boolean domain. After this step, the feature matrix X is transformed to a vector X containing respectively zeroes and ones for legitimate and phishing predictions. In Message Passing, the predictions are then passed through a traditional message passing GNN with h hidden layers, to propagate the information in the graph and learn node embeddings. A pooling method is used to reduce the dimension of node embeddings to a single node. Finally the resulting vector contains the probability of belonging into each class: phishing or benign.

Unlike other approaches, this solution takes advantage of the internal links structure of the website, along with the traditional features that led to successful results. Although the traditional GCN achieves great classification results, we see in embedding space that the model fails to delimit many nodes. However, the PRECLASSIFICATION step in PhishGNN, node embeddings are more grouped and classes can be delimited by a straight line, which leads to a better classifications. However in this approach, a graph is built from the HTML DOM and a GNN is fed with this graph. However, this method only relies on the HTML content, which could be easily stolen from benign websites in order to build perfect website copies. This method could thus be easily bypassed by cloning the HTML structure of legitimate websites.

Aman Rangapur, Tarun Kanakam and Dhanvanthini P [7] Phish-Defence, Phishing Detection Using Deep Recurrent Neural Networks. The proposed approach consists of two steps: Preparation of Dataset and Network Architecture and Training Parameters. For effective detection of malicious URLs, the dataset should contain recent URLs which are malicious, for recognizing fresh features to train the model. Attackers will change the

production of phishing links by anti-phishing regulations and procedures that have been released. Anti-phishing models and algorithms must also be improved based on new phishing data. Furthermore, the training dataset's quantity and validity significantly impact the performance of machine learningbased solutions. The performance of deep learning models increases with the variety of content in the training dataset. Hence, it is advised that phishing URLs and legitimate URLs should be extracted from data repositories. The dataset used in this paper consists of numerous legitimate and malicious URLs which are taken from PhishTank, OpenPhish, and Common Crawl. It consists of 46839 instances, and we merely looked into the text in the URL and extracted features to train the model. The dataset was split into 75% and 25% for training and testing, respectively. All the features captured from the URL are fed to the LSTM layer with an orthogonal recurrent initializer. The embedded matrix is used as weights, further combined with dense layers with a sigmoid activation function. A dropout of 0.5 is added between the LSTM layer and dense layer. Similarly, the features are captured from the URL and fed to the GRU layer with an orthogonal recurrent initializer and sigmoid recurrent activation, combined with dense layers with a softmax activation function. A dropout of 0.2 is added between the dense layers. Figure 5 and 6 depict the network architecture of PhishDefence of LSTM and GRU, respectively. The trained model is converted to a flite model for producing faster inference time on small edge devices like Raspberry Pi. The model is trained for 40 epochs with a batch size of 500 using the tanh activation function for the LSTM model and the sigmoid activation function for the GRU model. Binary cross entropy loss with Adam optimizer is attached to the dense layer for classifying legitimate and malicious URLs in both LSTM and GRU models.

The proposed method approaches produced the competitive classification accuracy of phishing websites among all other classifiers with all the feature selection methods used in this study with the lowest inference time. Thus, the proposed approach based on RNNs can also be used on smaller devices like RaspberryPi and Arduino to successfully predict phishing websites to contribute and provide more confidence for online commerce and business customers. The main limitations are the absence of comparisons between certain studies. Additionally, it wish to emphasize that while they properly presented the experiment findings from the studies they examined, they did not follow their experimental design or apply their models. There could be some missing materials. Therefore, it is difficult to say that this approach included all pertinent research.

Chenguang Wang, Yuanyuan Chen [8] proposed TCURL: Exploring hybrid transformer and convolutional neural network on phishing URL detection. In this paper they propose a hybrid network architecture, called TCURL, which considers both local and global correlations among the characters of URLs. TCURL has two parallel branches, a convolution branch and a transformer branch, and a fusion block used to deal with messages from the two branches. The convolution branch provides sufficient positional information meaning that no extra positional encoding is needed. Through the embedding process, a given URL is first transformed into a matrix with a shape of (L, C), which is then duplicated and fed into the two branches. Next a transformer decoder layer is used to fuse the outputs from the two branches. Finally, we flatten the output and employ a fully connected layer followed by a softmax activation to obtain the final result. TCURL represents a hybrid model designed to address the one-dimensional data binary classification problem. Experiments were designed and conducted to analyze the effect of the various elements in a hybrid transformer and CNN model. A dictionary of 67 unique characters (a valid placeholder, 26 lowercase letters, 10 digits, and 30 special characters) is used to convert the URL into a one-hot encoding matrix. The placeholder channel, which indicates whether the current character is valid, is initialized into ones. If the current position contains a valid character, we set the value of the placeholder channel to 0 and the value of this character channel to 1. Sixty-six character channels are initialized into zeros. We select 200 as the maximum length and discard any URLs with a length exceeding than 200. For example, the placeholder dimension of the URL www.google.com is 0...0111...1 which is composed of 14 zeros and 186 ones. Through the one-hot encoding, URLs can be represented in the form of 2-D matrices with the size of 200 × 67. Next, we use a 1-D convolution layer with a kernel size of 3 and stride of 1 to change the channel dimension into 64 channels. Finally, each URL is converted into a 200 × 64 matrix url_input through the embedding process. The algorithm used in this paper is a hybrid model that combines CNNs and Transformers for URL detection, with a focus on capturing both local correlations and long-term dependencies among characters in URLs.

Algorithm 1 The detection process of TCURL

Input: url

Output: detection_result

1: url_input = embedding(url)

2: trans_branch = url_input, conv_branch = url_input

3: for i = 0; i< N; i + + do

4: trans_branch = encoding_block(trans_branch)

5: end for

6: for j = 0; j < M; j + + do

7: conv_branch = analysis_block(conv_branch)

8: end for

9: fusion_output = fusion_block(trans_branch, conv_branch)

10: detection_result = softmax(project(fusion_result)).

In this approach they used three publicly available datasets called Phishing Site URLs (PSU), ISCX-URL2016, and Phishing and Benign Websites Dataset (PBWD) respectively. They employed five metrics to evaluate there model, namely accuracy (Acc), recall (Rec), false positive rate (FPR), precision (Pre), and the F1-score (F1). The result of the model is compared with the ground truth, and each URL is divided into true positive (TP), false positive (FP), true negative (TN) or false negative (FN).

The paper indicates that LayerNorm (Layer Normalization) exhibits clear advantages over other normalization methods, including BatchNorm (Batch Normalization), in the convolution branch. Consistency with Transformer Normalization Methods: The paper suggests that LayerNorm remains consistent with the normalization methods commonly used in transformers. This consistency is seen as beneficial because it facilitates the integration of outputs from both the convolution branch and the transformer branch in the output module. Suitability for Text Data: The paper mentions that LayerNorm is more suitable for text data, such as URLs, compared to BatchNorm. This suggests that LayerNorm performs well when dealing with the specific characteristics of text data.

Zainab Alshingiti, Rabeah Alaqel, Jalal Al-Muhtadi, Qazi Emad Ul Haq, Kashif Saleem and Muhammad Hamza Faheem [9] A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN. In this paper they have proposed three distinct approaches in order to train the data so that the output can be achieved efficiently. Numerous methods that assist in detecting phishing attacks have been applied by using different, new, and known features such as URL length, frequency of keywords, lexical features, and by incorporating new features.). The first step is data collection and

preprocessing. In this method they have used SelectK Best method is used in order to get optimised set. In training and testing step they have done it in three different ways they are

LSTM (long short-term memory) is a form of recurrent neural network (RNN) that gains superior results when dealing with time-series data, removing vanishing gradients and long-term dependencies. The architecture of LSTM is made up of a cell and three gates (input, output, and forget).A convolutional neural network is a kind of neural network that requires large, labelled data for training. CNNs play a significant role in many problems such as image classification, object recognition, phishing detection, and diagnosis of medical diseases. Input, convolution, pooling, and fully connected layers are the main layers needed to construct a CNN. Accelerating the learning process has led CNN to accomplish great and high results for many problems.LSTM–CNN architecture involves both CNN and LSTM methods in order to make use of the benefits of both methods and accomplish excellent performance. Since CNN and LSTM show high performance in overcoming classification, detection, and recognition tasks to using these three methods for the phishing detection task is promising.

The proposed approach has an advantage that CNN runs one order of magnitude faster than both LSTM and LSTM–CNN. The computations in CNNs can occur in parallel, in contrast to LSTM, which captures the dependency across time sequences in the input vector. However the approach has that the model does not check the status of the URL of the website, i.e., whether the website is active or not, which impacts the results. To overcome this limitation, it might be necessary to speed up the training process and improve feature engineering, which would then allow us to verify the website's state and improve training process accuracy.

Shouq Alnemari, Majid Alshammari [10] Detecting Phishing Domains Using Machine learning. This paper develops and compares four models for investigating the efficiency of using machine learning to detect phishing domains. It also compares the most accurate model of the four with existing solutions in the literature. These models were developed using artificial neural networks (ANNs), support vector machines (SVMs), decision trees (DTs), and random forest (RF) techniques. Moreover, the uniform resource locator's (URL's) UCI phishing domains dataset is used as a benchmark to evaluate the models. The proposed models were able to detect different types of attacks from the UCI dataset.

This dataset was created to build machine learning-based phishing website detection algorithms. It is comprised of extensive properties that span four distinct categories. They designed and extracted characteristics from the following categories: Address Bar, HTML and JavaScript, Abnormal, and Domain. This dataset has 11,055 records, and each record includes 31 characteristics. The characteristics of the collection are identified by names, such as URL Length, Submitting to Email, Shortening Service, Abnormal URL, Having an At Symbol, and Redirect. To increase accuracy, this paper utilized the Minmax normalization feature as a preprocessing step in each proposed model. Normalization is a useful strategy for improving the accuracy of machine learning models, and it is required for some models to work properly. The Minmax normalization technique in the suggested model compresses the data to a domain of [0, 1], which improves the model training input quality

$$X\_std = (X - X.min)/(X.max - X.min) \quad \text{.....(1)}$$

$$X\_scalar = X\_std \times (max - min) + min \quad \text{.....(2)}$$

The UCI dataset is split 80/20 into training and testing sets, respectively, by using c5-fold cross-validation, which presented the best performance in the latest research. Machine learning algorithms can learn to identify phishing websites based on their features rather than relying on predefined rules or signatures. This makes them more robust and less prone to false positives or false negatives. Another advantage of using machine learning for phishing detection is that it can be easily integrated into existing security systems and workflows. However it requires features of the URL to be manually extracted which depends on third-party services to obtain certain important features. It doesn't cope up with huge volumes of data and URL doesn't confirm to manually constructed features.

Eman Abdullah Aldakheel , Mohammed Zakariah , Ghada Abdalaziz Gashgari , Fahdah A. Almarshad and Abdullah I. A. Alzahrani [11]A Deep Learning-Based Innovative Technique for Phishing Detection in Modern Security with Uniform Resource Locators. We evaluate the performance of our model on the PhishTankdataset, which is a widely used dataset for detecting phishing websites based solely on Uniform Resource Locators (URL) features. Binary-categorical loss and the Adam optimizer are used, the accuracy of the k-nearest neighbours (KNN), Natural Language Processing (NLP), Recurrent Neural Network (RNN), and Random Forest (RF) models is 87%, 97.98%, 97.4% and 94.26%, respectively.1D CNN Architecture is used as Model Architecture. A 1D CNN model is a

CNN model that only has one dimension, such as text or time series data. An input layer, one or more convolutional layers, pooling layers, and an output layer make up the fundamental building blocks of a 1D CNN model. Input layer receives input data, typically preprocessed and transformed into numerical representations such as tokenized text or time series data. The pooling layers of a 1D CNN model are responsible for lowering the feature dimensionality. The output layer of a 1D CNN model produces the final prediction.1D CNNs can be trained using the SGD back-propagation algorithm or other optimization algorithms such as Adarad, Adam, and others.

The proposed method for detecting phishing sites using DL has the potential to be implemented in real-world environments to improve cybersecurity. The method employs a CNN for high-accuracy classification of URLs to distinguish between genuine and phishing site. However this approach has only been evaluated on the PhishTank dataset, which is widely used but may not be representative of all types of phishing attacks. This dataset bias could limit the generalizability of the proposed approach. Additionally, the proposed approach requires crawling and analyzing URLs, which may not be suitable for the real-time detection of phishing attacks. This could limit the applicability of the proposed approach in certain situations where real-time detection is critical.

Manoj Kumar Prabakaran, Parvathy Meenakshi Sundaram, Abhinaya Devi Chandrasekar [12] have proposed a enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders. This approach employs OHE(One-hot-encoding) based preprocessing mechanism that converts every URL string into a numerical vector with N x M dimension. M denotes the length of the maximum number of probable characters that might appear in a URL.N denotes the length of the URL, that is, the number of characters in the URL. After preprocessing the URL inputs, instead of directly feeding them to a neural network-based model for classification, our approach adopts a feature reduction/extraction technique to select certain inherent features of a URL to optimise the performance of the classifier. In order to automatically extract salient features from the input URL vector, They developed an Autoencoder (AE)-based feature extraction approach. The Auto encoders is a special form of feed forward neural network which is mainly designed to encode the input into a compressed and meaningful representation and then decode it back such that the

reconstructed input is similar as possible to the original one. This model suffers from the problem of generalisation.

To avoid the problem of overfitting and non-regularisation in the latent space of a traditional AE model, have adopted a Variational Autoencoder (VAE), a special form of AE model. This is capable of learning smooth latent space representation of the input data. VAE can be an effective tool, however it has some drawbacks. The encoding and decoding process can lead to information loss, resulting in blurry and low-resolution samples that lack the sharpness and detail of the real data. Additionally, VAE can suffer from posterior collapse, where the encoder ignores the input data and outputs a trivial latent space, leading to poor representation and reconstruction. It is also sensitive to the choice of the prior distribution, the likelihood function, and the regularization term, which can affect the trade-off between the fidelity and the diversity of the model.

1.2 Problem Statement

In recent years, Internet technologies are grown pervasively not only in information-based web pages but also in online social networking and online banking, which made people's lives easier. As a result of this growth, computer networks encounter with lots of different security threats from all over the world. Malicious URLs are a common threat to the Web users. Phishing aims at stealing sensitive information by fooling victims with falsified interfaces. In the case of phishing websites, attackers usually try to impersonate well-known and widely used services such as social media, banks and e-commerce websites. Such spoofed websites are often built from the same code base as the original site, which could make them difficult to detect at first glance. Most phishing URLs tend to be very long, with multiple sub-domains and special characters. Developed and deployed an LSTM neural network and Dense model that leverages deep learning techniques and real-time monitoring to detect and mitigate phishing activities effectively.

1.3 Objectives

- The objective of this project is to train deep neural network on the dataset created to predict phishing websites.
- To recognize the legitimate websites and phished websites through URL.
- Implementation of Features Based Extraction of URL (Uniform Resource Locator) so as to decrease manual extraction.

- Deep learning training model that recognizes both the legitimate and phishing websites.
- Implementing the model to directly recognize the phishing website in the browser.

1.4 Scope of the Project

- A deep learning model named LSTM and Dense layer is used for phishing websites detection.
- It detects 30 features from url. If website is resulted as phished then a page will be displayed which shows it as phished.
- It blocks the phished website without further procedures.
- This project is deployed for single table application.

1.5 Organization of the Report

The chapter 1 includes introduction to anti-phishing website detection along with literature survey, problem statement, objectives, scope of the project. Chapter 2 brings out domain specific of the project. Chapter 3 discusses about system design and implementation later in Chapter 4 gives the Result and snapshots of the project. Finally, Chapter 5 concludes the report along with possible future enhancements for this project followed by references.

CHAPTER 2

DEEP LEARNING ALGORITHMS

The details of the methodology of the Anti-Phishing website detection. The below information shows about deep learning algorithms and how it's implemented in the present system.

2.1 Deep Learning

Deep learning can be defined as the method of machine learning and artificial intelligence that is intended to intimidate humans and their actions based on certain human brain functions to make effective decisions. It is a very important data science element that channels its modeling based on data-driven techniques under predictive modeling and statistics. To drive such a human-like ability to adapt and learn and to function accordingly, there have to be some strong forces which we popularly called algorithms. Deep learning algorithms are dynamically made to run through several layers of neural networks, which are nothing but a set of decision-making networks that are pre-trained to serve a task. Later, each of these is passed through simple layered representations and move on to the next layer. However, most machine learning is trained to work fairly well on datasets that have to deal with hundreds of features or columns. For a data set to be structured or unstructured, machine learning tends to fail mostly because they fail to recognize a simple image having a dimension of 800x1000 in RGB. It becomes quite unfeasible for a traditional machine learning algorithm to handle such depths.

Deep learning can be used for supervised, unsupervised as well as reinforcement machine learning. it uses a variety of ways to process these.

Supervised Machine Learning: Supervised machine learning is the machine learning technique in which the neural network learns to make predictions or classify data based on the labeled datasets. Here we input both input features along with the target variables. the neural network learns to make predictions based on the cost or error that comes from the difference between the predicted and the actual target, this process is known as backpropagation. Deep learning algorithms like Convolutional neural networks, Recurrent neural networks are used for many supervised tasks like image classifications and recognization, sentiment analysis, language translations, etc.

Unsupervised Machine Learning: Unsupervised machine learning is the machine learning technique in which the neural network learns to discover the patterns or to cluster the dataset based on unlabeled datasets. Here there are no target variables. while the machine has to self-determined the hidden patterns or relationships within the datasets. Deep learning algorithms like autoencoders and generative models are used for unsupervised tasks like clustering, dimensionality reduction, and anomaly detection.

Reinforcement Machine Learning: Reinforcement Machine Learning is the machine learning technique in which an agent learns to make decisions in an environment to maximize a reward signal. The agent interacts with the environment by taking action and observing the resulting rewards. Deep learning can be used to learn policies, or a set of actions, that maximizes the cumulative reward over time. Deep reinforcement learning algorithms like Deep Q networks and Deep Deterministic Policy Gradient (DDPG) are used to reinforce tasks like robotics and game playing etc.

2.1.1 LSTM (Long Short Term Memory)

Long Short-Term Memory is an improved version of recurrent neural network designed by Hochreiter & Schmid Huber. LSTM is well-suited for sequence prediction tasks and excels in capturing long-term dependencies. Its applications extend to tasks involving time series and sequences. LSTM's strength lies in its ability to grasp the order dependence crucial for solving intricate problems, such as machine translation and speech recognition. The article provides an in-depth introduction to LSTM, covering the LSTM model, architecture, working principles, and the critical role they play in various applications.

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well-suited for tasks such as language translation, speech recognition, and time series forecasting. LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.

LSTMs Long Short-Term Memory is a type of RNNs Recurrent Neural Network that can detain long-term dependencies in sequential data. LSTMs are able to process and

analyse sequential data, such as time series, text, and speech. They use a memory cell and gates to control the flow of information, allowing them to selectively retain or discard information as needed and thus avoid the vanishing gradient problem that plagues traditional RNNs. LSTMs are widely used in various applications such as natural language processing, speech recognition, and time series forecasting.

The main idea of how LSTM works is that instead of using the same feedback loop connection at each input sequence, it uses 2 separate paths to make new predictions. One path is for long-path memory and the other is for short-path memory. The Long - Term memory also called the cell state that does not include trainable weights and biases. Hence, it can flow throughout the input sequence without running the risk of vanishing gradients. The hidden state is represented by the short-term memory path and it's connected to a set of weights.

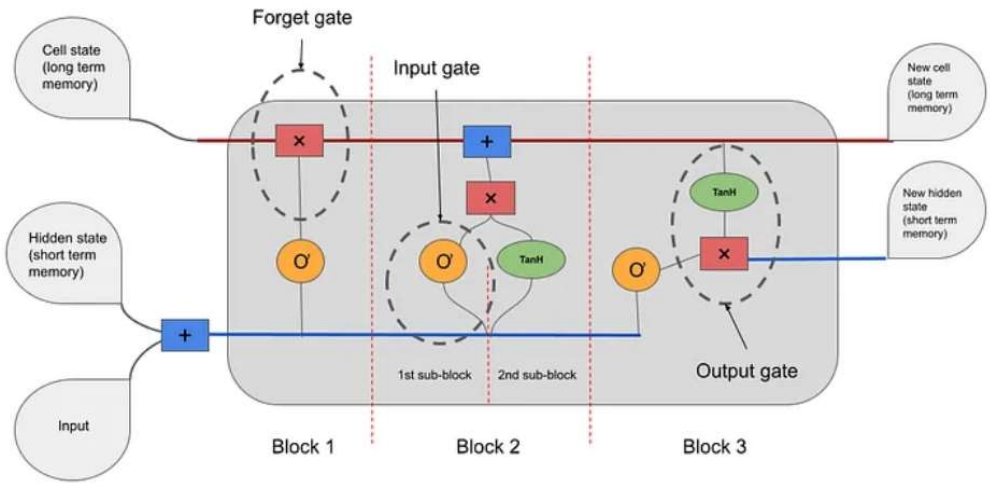


Fig 2.1 Unit architecture of LSTM

Fig 2.1 shows, the internal architecture of an LSTM unit and the workflow within a single unit.

The first building block of the LSTM unit determines the percentage of the Long-Term Memory (cell state) that will be retained. Technically it's called the forget gate. This block combines the input with the short-term memory and then runs the resulting computations through a Sigmoid activation function. It returns a number between 0 and 1 representing the percentage of the long-term memory that will be used in the following steps.

The second building block of the LSTM unit consists of 2 sub-blocks. The second sub-block combines the same input with the short-term memory (hidden state) to create

a potential Long term memory using different trainable weights. This combination takes the form of a weighted sum of the input and the short-term memory and then runs the result through a Tanh activation function that returns a number between -1 and 1.

But, how much of this potential long-term memory will be added to the actual long-term memory value is controlled by the first sub-block. The first sub-block performs the same mathematical combination between the input and the short memory term but the resulting output will pass through a Sigmoid activation function. The final result represents the percentage of how much of the future potential long-term memory value, will be added to the actual cell state.

After updating the long-term memory during the first 2 blocks, the final block is responsible for refreshing the short-term memory. It gets the new value from the forget gate and passes it to the tank activation function to produce the potential short-term memory. Then, following the same steps as in the second block, a second sub-block will combine the input and the old short-memory term to produce the percentage of how much the unit will retain from the potential value. This block results in a new hidden state and a new short-term memory representing the final output of the LSTM unit also called the output gate.

To wrap, a single LSTM contains three fundamental parts also called gates. The first gate or the forget gate and the input gate update the cell state with respect to the current input. The output gate returns the new hidden state based on the newly calculated long-term memory or the cell state.

The advantages of LSTM (Long-Short Term Memory) are as follows:

- Long-term dependencies can be captured by LSTM networks. They have a memory cell that is capable of long-term information storage.
- In traditional RNNs, there is a problem of vanishing and exploding gradients when models are trained over long sequences. By using a gating mechanism that selectively recalls or forgets information, LSTM networks deal with this problem.
- LSTM enables the model to capture and remember the important context, even when there is a significant time gap between relevant events in the sequence. So where understanding context is important, LSTMS are used. eg. machine translation.

2.1.2 Dense Layer

A dense layer, also known as a fully connected layer, is a fundamental building block in neural networks where every neuron in the layer is connected to every neuron in the previous layer. This layer performs a weighted sum of the inputs, adds a bias term, and applies an activation function to introduce non-linearity. This layer is the most commonly used layer in artificial neural network networks. The dense layer's neuron in a model receives output from every neuron of its preceding layer, where neurons of the dense layer perform matrix-vector multiplication. Matrix vector multiplication is a procedure where the row vector of the output from the preceding layers is equal to the column vector of the dense layer. The general rule of matrix-vector multiplication is that the row vector must have as many columns like the column vector. The primary advantage of dense layers is that they are able to capture complex patterns in data by allowing each neuron to interact with all the neurons in the previous layer. This makes dense layers well-suited for tasks such as image classification, where the input is a high-dimensional image, and the output is a prediction of the class of the object in the image. However, the downside of dense layers is that they require many parameters, making them computationally expensive to train. In addition, dense layers can suffer from overfitting, where the model memorizes the training data instead of learning generalizable patterns. In an LSTM (Long Short-Term Memory) network, the dense layer typically refers to the fully connected layer that follows the LSTM layers. This layer takes the output of the LSTM cells and processes it further, often transforming it into the desired output format, such as class probabilities in a classification task or continuous values in a regression task.

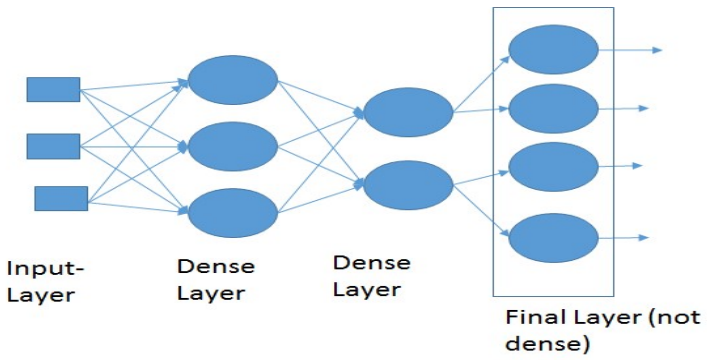


Fig 2.2 Dense Layer

The types of Dense layer are:

Single Layer Perceptron: The perceptron is a single processing unit of any neural network. Frank Rosenblatt first proposed in 1958 is a simple neuron which is used to classify its input into one or two categories. Perceptron is a linear classifier, and is used in supervised learning. It helps to organize the given input data. A perceptron is a neural network unit that does a precise computation to detect features in the input data. Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as Linear Binary Classifier. The perceptron consists of 4 parts.

Input value or One input layer: The input layer of the perceptron is made of artificial input neurons and takes the initial data into the system for further processing.

Weight: It represents the dimension or strength of the connection between units. If the weight to node 1 to node 2 has a higher quantity, then neuron 1 has a more considerable influence on the neuron.

Bias: It is the same as the intercept added in a linear equation. It is an additional parameter which task is to modify the output along with the weighted sum of the input to the other neuron.

Net sum: It calculates the total sum.

Activation Function: A neuron can be activated or not, is determined by an activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.

Multi-Layer Perceptron: Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons. A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. In multi layer perceptron there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer. Every node in the multi-layer perception uses a sigmoid

activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.

2.2 Selenium

Selenium Python bindings provides a simple API to write functional/acceptance tests using Selenium WebDriver. Through Selenium Python API you can access all functionalities of Selenium WebDriver in an intuitive way.

Selenium WebDriver is a popular open-source library and a key component of the Selenium automation framework used to automate testing for web applications. It is a collection of APIs which leverages a programming interface for developers and testers to write scripts in various programming languages such as Java, JavaScript, C#, Python, etc. to automate web browser's action and retrieve information from web pages.

Through test scripts, WebDriver simulates user actions, navigates through web pages, interacts with elements (such as button, text, dropdown menu, forms, links, etc), submit forms, perform validations, assertions and many more. As per Selenium document "WebDriver drives a browser natively, as a user would, either locally or on a remote machine using the Selenium server, marks a leap forward in terms of".

The first step for you to start working on Selenium with Python is that need to write functional test cases using the Selenium web driver. Then, to forward a request to the Selenium server that sits at the back-end, which will execute the test cases on the browsers automatically. Then can perform tests on any browser including Firefox, Chrome, IE, etc. The only thing that changes are the web drivers, which are specific for each browser.

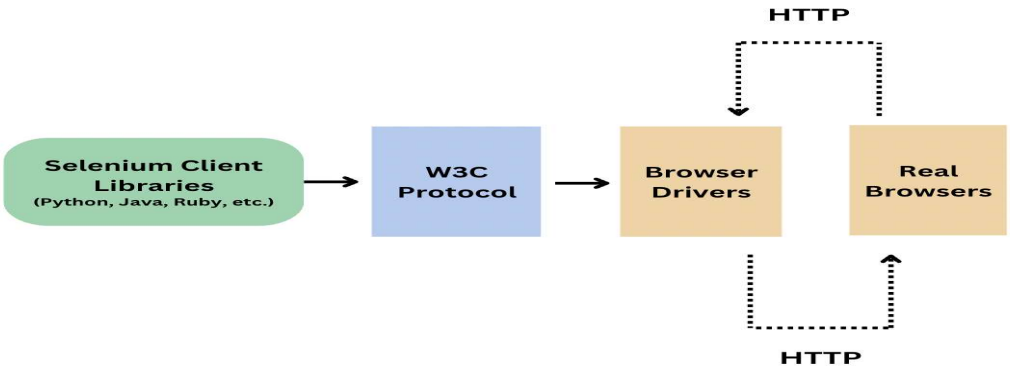


Fig 2.3 : Selenium Architecture

Selenium 4 uses W3C protocol instead of JSON wire protocol for communication between Client Libraries and Browser Drivers.

Selenium WebDriver Architecture is made up of four major components:

- Selenium Client library: Selenium provides support to multiple libraries such as Ruby, Python, Java, etc as language bindings
- W3C : It stands for the World Wide Web Consortium, an international community that develops and maintains standards and guidelines for the World Wide Web. The main aim of the W3C is to ensure the long-term growth and interoperability of the
- Web Browser Drivers: Selenium browser drivers are native to each browser, interacting with the browser by establishing a secure connection. Selenium supports different browser drivers such as ChromeDriver, GeckoDriver, Microsoft Edge WebDriver, SafariDriver, and InternetExplorerDriver.
- Browsers: Selenium provides support for multiple browsers like Chrome, Firefox, Safari, Internet Explorer etc.

Selenium requires a web driver, which will help it interface with the browser that you want to run your tests on. A ChromeDriver is a separate executable or a standalone server that Selenium WebDriver uses to launch Google Chrome. Here, a WebDriver refers to a collection of APIs used to automate the testing of web applications. As Google Chrome dominates the browser market, the use of a ChromeDriver becomes a must. Selenium WebDriver uses the ChromeDriver to communicate test scripts with Google Chrome. It is used to navigate between web pages and provide input to the same.
ChromeDriver is a separate executable that Selenium WebDriver uses to control Chrome. It is maintained by the Chromium team with help from WebDriver contributors.
Step to setup tests for running with ChromeDriver:
Ensure Chromium/Google Chrome is installed in a recognized location
ChromeDriver expects you to have Chrome installed in the default location for your platform. You can also force ChromeDriver to use a custom location by setting a special capability.
Download the ChromeDriver binary for your platform under the downloads section of this site
Help WebDriver find the downloaded ChromeDriver executable.

A ChromeDriver is a separate executable or a standalone server that Selenium WebDriver uses to launch Google Chrome. Here, a WebDriver refers to a collection of APIs used to automate the testing of web applications.
Initializing the object of ChromeDriver is possible with the help of this command:
WebDriver driver = new ChromeDriver

Advantages of Selenium

There are multiple reasons as to why Selenium is way more popular and powerful when compared to other web-automation tools in the market. Here are the reasons why:

- It is an open-source, free-to-use, portable tool that is used to perform web-testing in a seamless manner.
- It is a combination of several Domain Specific Language (DSL) and other tools which allow you to perform various types of testing.
- It is quite easy to understand, intuitive, and has a low learning curve. The commands are categorized as multiple classes, making them easier to understand.
- It takes off a load of the burden from testers, since the amount of time that is required to perform testing, especially on repeated test cases reduces drastically.
- There is a significant reduction in the costs incurred to the business clients, which is a win-win situation.

2.3 Summary

The chapter 2 includes Domain Specific which explains the deep learning deep learning algorithms and also Selenium which are used for the detection of phishing website. Next chapter system design and implementation is discussed.

CHAPTER 3

SYSTEM DESIGN AND IMPLEMENTATION

The chapter 3 defines the workflow of the system design, implementation of the model and how the system is designed step by step in detail.

3.1 System Design

The main aim of this project is to check whether the provided url is benign or phished by considering some of the features of the url.

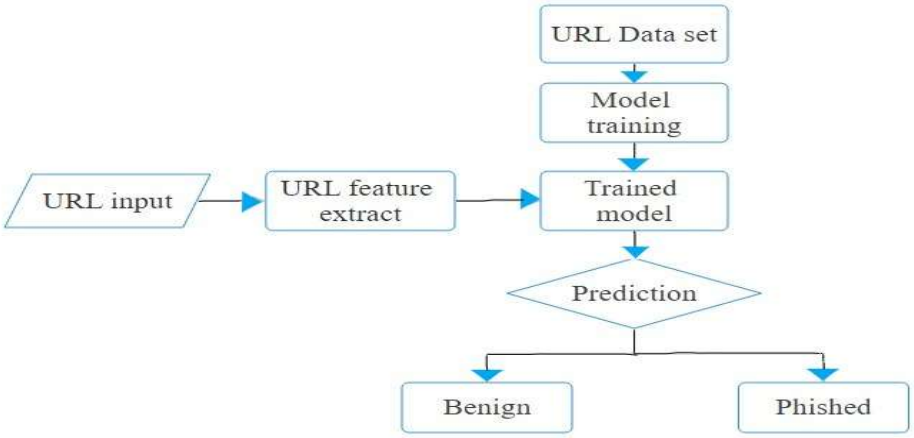


Fig 3.1 System Framework

The fig 3.1 depicts the work flow of the system describing url taken by user in browser, extracting the features from the url, giving the extracted features to the trained model and predicting on the basis of estimation score of the url whether it is benign or phished.

1. **Data Collection:**

URL data collection is a crucial initial step in building a phishing detection system using deep learning. Dataset was collected from UCI and it has 11055 lists of websites where there are both legitimate and phishing websites. A URL consists of 5 basic components First part of the URL is called Protocol which is a set of rules used for transferring data. Second part of the URL is called domain which has several parts. First part of domain often contains “www” prefix which referred as World Wide Web Address. Then, the name of the website is given after a dot. After a dot, the organization type will

be added which is followed by the country code if added. Third part of URL contains the path details which addresses the section and page of the website. Fourth part consists of query which refers part of a page. Finally the Query part sends additional information with the page request.

2. Feature Extraction:

Basically, there are 4 main features which has in total 30 sub-features. Based on the data, every single feature provides information about whether the website can be legit, phishing or suspicious. Address based feature generally shed light on the name of the URL. It has 12 sub-features those were IP address, URL length, Tiny URL, having @ symbol in url, using // symbol, having - in domain name, dots in domain, HTTPs, domain expiry date. Abnormal based feature has 6 sub-features. It generally focuses on unusual activities on the website those are Using <a>tags, Request URL, Links in <meta>, <script>and <Link>tag, Server Form Handler (SFH), Submitting Information to Email.HTML and JavaScript based 5 sub-features are Website forwarding, Status Bar Customization, Disabling Right Click, Pop-up Window, IFrame usage. Domain based 7 sub-features are Age of Domain, DNS record, website traffic, page rank, google index, links pointing to page, statistical reports.

3. Data Pre-processing :

Lays the foundation for model training by loading the dataset from a CSV file using Pandas' read_csv() function. While missing data handling isn't explicitly addressed, techniques like imputation or removal could be applied. Features and labels are then extracted, with feature selection or extraction techniques left implicit. The dataset is split into training and testing sets using scikit-learn's train_test_split() function, ensuring a robust evaluation of model performance. Although scaling or normalization isn't included, it's crucial for standardizing feature scales.

4. Data Building and Training:

Data preprocessing precedes the training of a Long Short-Term Memory (LSTM) neural network model aimed at detecting phishing websites. The LSTM architecture, defined using Keras Sequential API, comprises two LSTM layers with 128 and 64 units, followed by Dense layers for output. Following compilation with the Adam optimizer and Mean Squared Error loss function, training is executed using the fit() function on the training data, employing a batch size of 1 and 9 epochs. The model's performance can be assessed using various regression metrics on the test set. Post-training, the trained model is saved to a file named 'phishing.h5' for future use. Through these steps, the LSTM

model acquires the capability to distinguish between phishing and legitimate websites based on the provided features.

5. Model Evaluation and Validation:

Upon completing the training of the LSTM model, its performance is rigorously evaluated and validated to ensure reliability and effectiveness in detecting phishing websites. The evaluation process involves splitting the dataset into training and testing sets, allowing the model to be tested on previously unseen data. Key performance metrics such as accuracy, precision, recall, and F1 score are calculated to assess the model's ability to correctly identify phishing sites while minimizing false positives and false negatives. Cross-validation techniques are also employed to further validate the model's generalization capability across different subsets of the data. During validation, the model's predictions are compared against the actual labels to determine its predictive accuracy. Any discrepancies or patterns of misclassification are analysed to refine the model further.

6. Model Optimization and Fine-Tuning:

In this phase, the model's parameters and architecture are fine-tuned to improve its performance. This involves selecting the optimal architecture, including the number of layers and units, and tuning hyperparameters like learning rate, batch size, and epochs through techniques such as grid search. Regularization methods, like dropout, and early stopping are employed to prevent overfitting. Additionally, the activation functions and optimization algorithms are refined to enhance model accuracy and convergence speed. These adjustments ensure the model generalizes well and performs effectively.

7. Model Deployment:

Model deployment involves integrating the trained LSTM model into a real-time phishing detection system. This is achieved by embedding the model within a web application that utilizes Selenium's WebDriver to monitor user browsing activity. The system continuously extracts features from visited URLs and feeds them into the model for prediction. Upon detecting a phishing threat, the system takes immediate action, such as alerting the user or blocking access to the suspicious site.

3.2 Implementation

The implementation includes steps like Feature extraction, Training, Testing, Chrome check, comparing it and considering best model and predicting results.

1. **Feature Extraction:**

   Import necessary libraries

   Define a function having_IPhaving_IP_Address(url):

   Use regular expression to search for IP address patterns in the URL

   If a match is found, return -1 indicating presence of IP address, else return 1

   Define a function URLURL_Length(url):

   Check the length of the URL

   Return 1

   if length <= 75, 0 if 54 < length <= 75, else return -1

   Define similar functions for the remaining features:

   - Shortining_Service(url)
   - having_At_Symbol(url)
   - double_slash_redirecting(url)
   - Prefix_Suffix(url)
   - having_Sub_Domain(url)
   - SSLfinal_State(url)
   - Domain_registeration_length(url)
   - Favicon(url)
   - port(url)
   - HTTPS_token(url)
   - Request_URL(url)
   - URL_of_Anchor(url)
   - Links_in_tags(url)
   - SFH(url)
   - Submitting_to_email(url)
   - Abnormal_URL(url)
   - Redirect(url)
   - on_mouseover(url)
   - RightClick(url)

- popUpWidnow(url)
- Iframe(url)
- age_of_domain(url)
- DNSRecord(url)
- web_traffic(url)
- Page_Rank(url)
- Google_Index(url)
- Links_pointing_to_page(url)
- Statistical_report(url)

Define a main function main(url):

Call each of the feature-checking functions with the provided URL as argument

Store the results of each check in a list

Print the list of results

Return the list of results.

2. Training:

Read the url feature extracted files by using read.csv() of pandas library

Normalise it by summing up

if there are any missing values in the dataset by checking isnull()

for suming up sum()

Extract input features and store it in x

Split the dataset into training and testing sets along with specifying the size.

Define the LSTM model architecture

Initializing a Sequential model

model = Sequential()

model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))

for adding LSTM layer with 128 units

return sequences set to True

model.add(LSTM(64, return_sequences=False))

for adding another LSTM layer with 64 units

return sequences set to False

model.add(Dense(25))

for adding a Dense layer with 25 units

Display model summary to visualize the model architecture

Compile the model with adam optimizer and loss function MSE

Train model using fit() function on training dataset

Save () function.

3. Testing:

Function load_model(fileModelJSON, fileWeights):

a. Open fileModelJSON and parse its content as JSON, storing it in model_json.

b. Create a neural network from JSON representation using model_from_json.

c. Load the weights of the model from fileWeights.

d. Return the loaded model.

Call load_model with arguments "deeplearning_LSTM.json" "deeplearning_LSTM.h5":

a. Assign the returned model to the variable model.

b. Obtain the layers of the model and store them in l_layers.

c. Extract the weights of the first layer and assign them to the variable weights.

d. Display the shape of the weight matrix.

Define test_url_mal and test_url_benign as strings representing URLs.

Assign test_url_mal to the variable url.

Define max_len as 75.

Convert the characters of the URL into integers based on positions in "printable" string:

a. Iterate through each character x in the URL.

b. If x is a printable character:

i. Get its index in the "printable" string and add 1.

ii. Store the result in a list of lists url_int_tokens.

Pad or truncate the list of lists url_int_tokens to have a maximum length of max_len, storing the result in X.

Use the model to predict the target probability for X:

a. Call model.predict with X as input and batch_size=1.

b. Store the result in target_proba.

Define a function named print_result that takes proba as input:

a. If proba is greater than 0.5, return "phishing".

b. Otherwise, return "normal".

Print the test URL and its predicted result:

a. Print "Test URL.

4. Chrome Check:

Import necessary libraries and modules

Define a function to load the phishing detection model

Load the phishing detection model from a saved file ('phishing.h5')

Initialize a Chrome webdriver

Set the URL of the website to "https://www.google.com/"

Open the webdriver and navigate to the specified URL

While True:

Get the current URL of the webdriver

Print the current URL

If the current URL is different from the initial URL:

Print "entered phishing detection"

Call the 'main' function from 'inputScript' module to extract features from the URL

Store the extracted features in a variable

Preprocess the features and convert them into a numpy array

Print "Phishing Website Detection"

Predict whether the website is phishing using the loaded model

If the prediction score is less than 0.93:

Open a localhost URL indicating a phishing website detection alert

Update the URL variable to the current URL

Print "phishing website"

3.3 Summary

In Chapter 3 the design of the system is discussed in detail of the phishing website detection system design and its implementation. Along with this software and deep learning algorithm that is being used is also discussed. The next chapter contains results and snapshots.

CHAPTER 4

RESULT AND SNAPSHOT

The chapter 4 defines the results and analysis of phishing website detection presented. In previous chapters, the implementation techniques are discussed. In this chapter the performance of proposed model.

4.1 Result Analysis

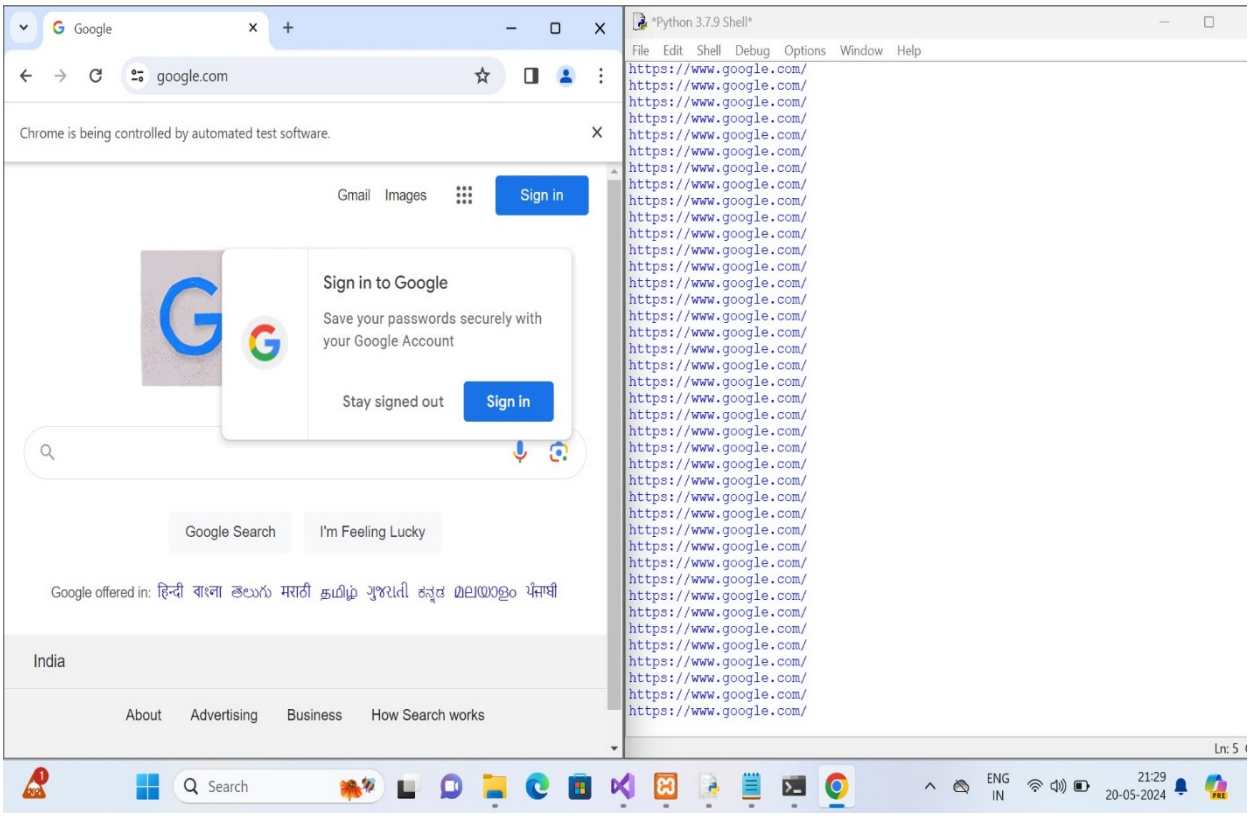


Fig 4.1 Snapshot of Default Chrome browser in both frontend and backend

In the fig 4.1 The chrome is being controlled by automated testing software. Selenium, a web automation tool, is used to interact with a web browser (Google Chrome). The script opens a Chrome browser window and navigates to a predefined URL.

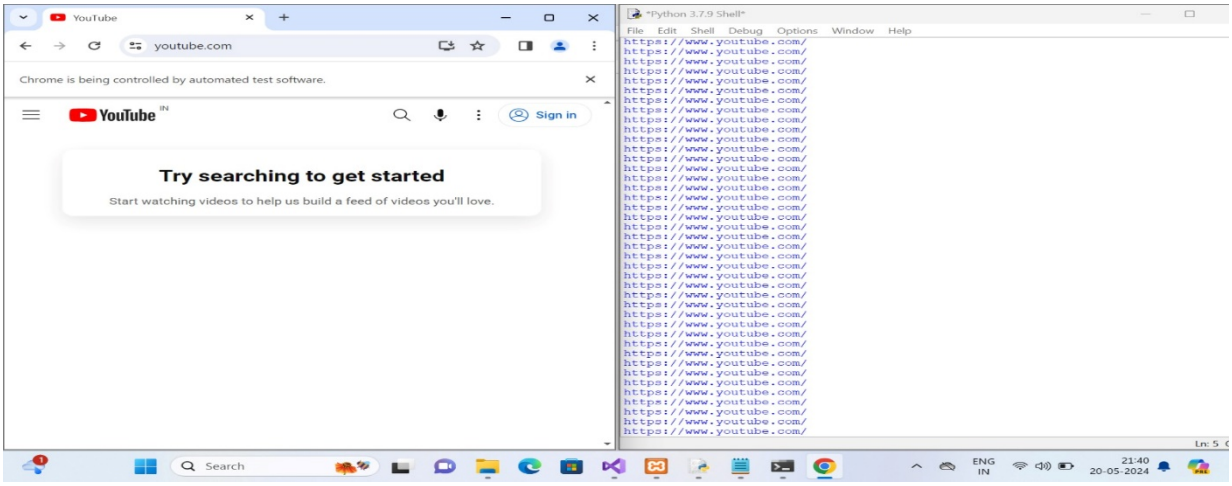


Fig 4.2 Snapshot of Genuine website (youtube.com)

In the fig 4.2 The script enters a continuous loop (while(1)) where it continuously monitors the current URL in both frontend and backend.

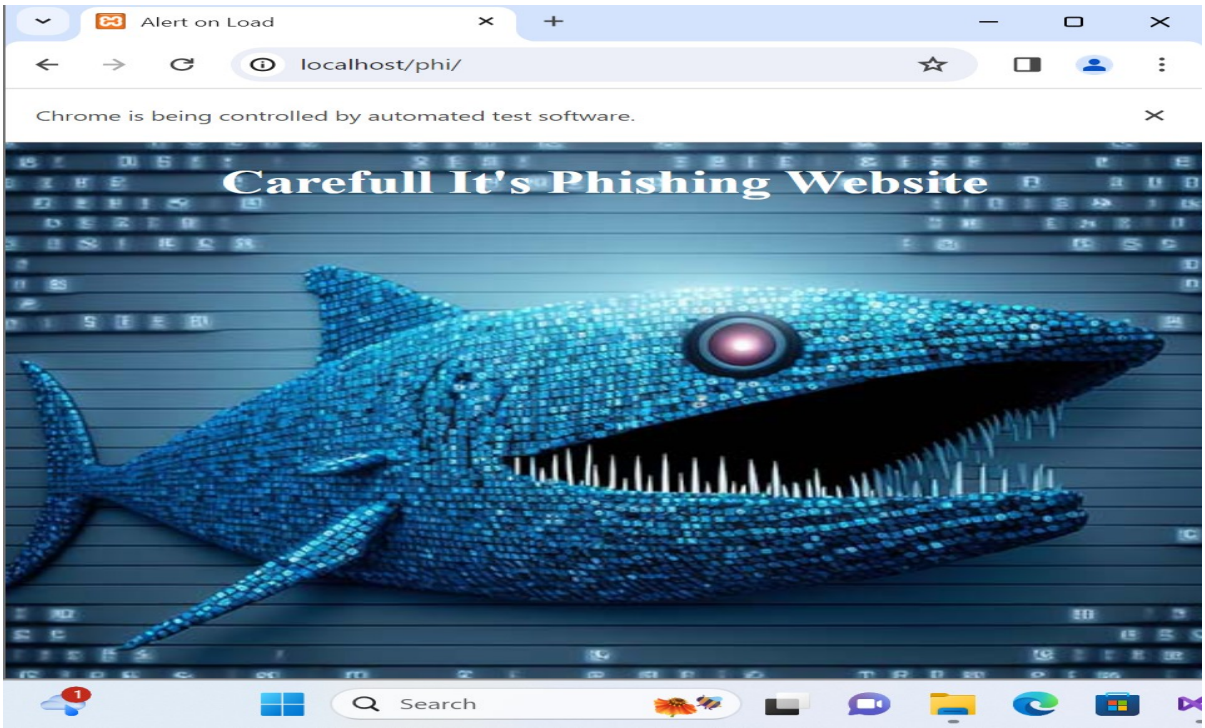


Fig 4.3 Snapshot of phished website

In the fig 4.3 If a change in the URL is detected, indicating a potential phishing attempt, the script initiates the phishing detection process.

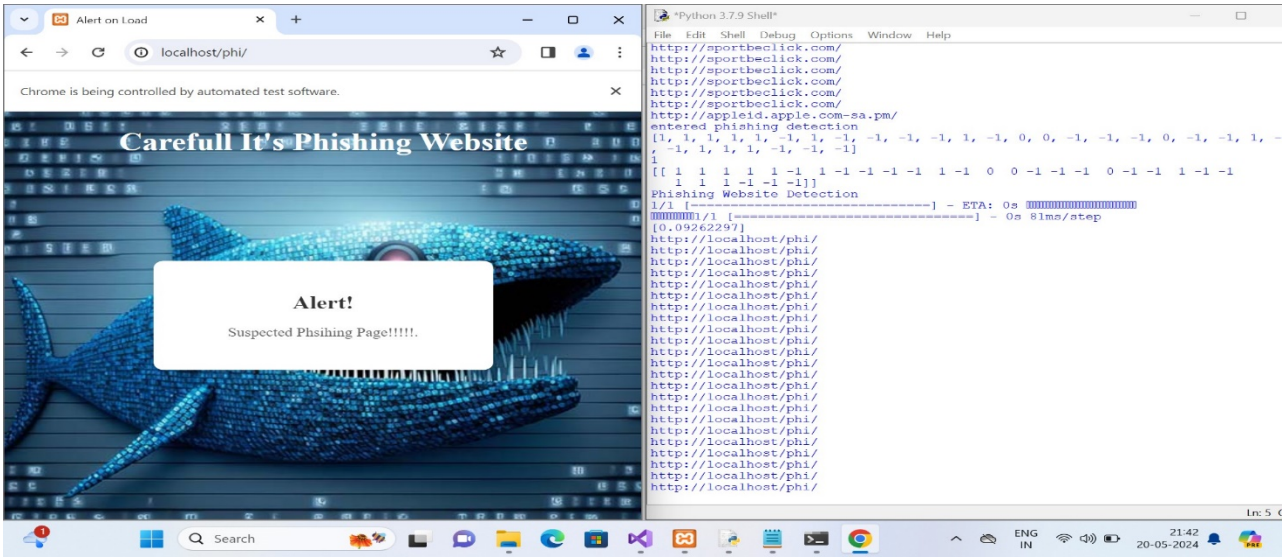


Fig 4.4 Snapshot of Popup message

In the fig 4.4 If the predicted probability of phishing is below a certain threshold (0.96), it takes action by redirecting the user to a predefined phishing alert page (http://localhost/phi/). This action serves as a warning to the user about the potential phishing threat.

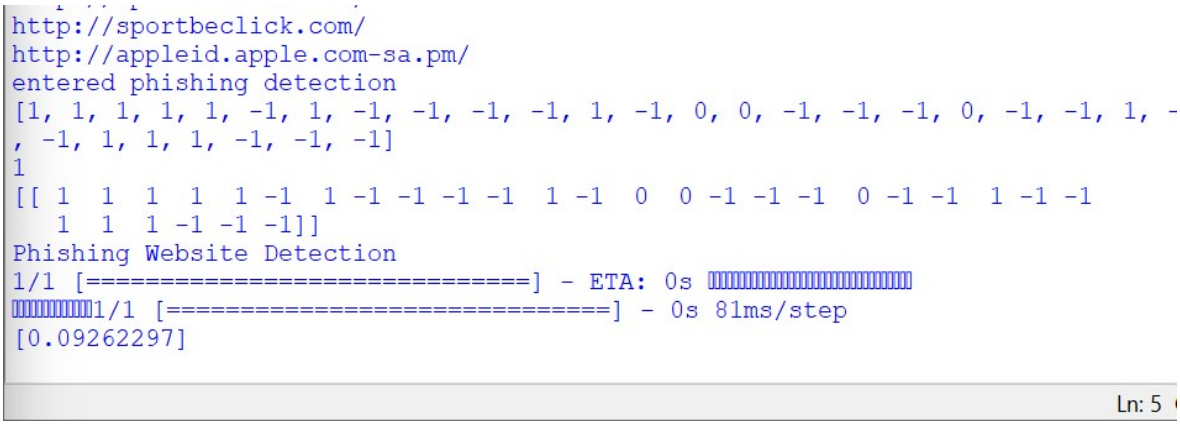


Fig 4.5 Snapshot of Feature analysis

In the fig 4.5 it provides feature analysis of detected phished website.

4.2 Summary

In Chapter 4 the result analysis and the snapshot of results is discussed. The next chapter contains conclusion.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

Anti-phishing measures are critical in safeguarding users and organizations against malicious attempts to steal sensitive information. Through a combination of technological solutions, user education, and proactive detection methods, the risks posed by phishing attacks can be mitigated effectively. Technologies like deep learning models promising avenues for developing advanced anti-phishing systems capable of detecting and preventing sophisticated phishing attempts. Using LSTM got an accuracy which effective than other two model. Deployment is done directly on browser using selenium module. By staying vigilant, adopting best practices, and continuously innovating anti-phishing measures, individuals and organizations can reduce the likelihood of falling victim to phishing attacks and protect their digital assets and privacy.

5.2 Future Scope

- Integration with cloud storage allows for large-scale data storage and processing, facilitating the training of deep learning models on vast amounts of labelled and unlabelled phishing data.
- The model can be deployed into the chrome extension so that every user across the world can make use of it.

PUBLICATION

[1] Akshatha A P, Chaithra R, Madhura S, Chandana C Sagar, Hiriyanna G S, "Literature Review on Phishing Website Detection Using Deep Learning", International Journal of Scientific Research in Engineering and Management (IJSREM), Volume: 08 Issue: 05 | May - 2024.

[2] Akshatha A P, Chaithra R, Madhura S, Chandana C Sagar, Hiriyanna G S, "Anti-Phishing: A Web Identifier For Spoofed Sites Using Neural Network", International Journal of Scientific Research in Engineering and Management (IJSREM), Volume: 08 Issue: 05 | May – 2024 | DOI: 10.55041/IJSREM34558.

REFERENCES

[1] Orunsolu Abioduna,Sodiya A.Sb, Kareem S.O, "Link Calculator –an efficient link-based phishing detection tool", Acta Informatica Malaysia, Volume 4, Issue 2, September 2020, p. 37-44.

[2] Md.Faisal Khana, B L Rahab, "Detection of Phishing Websites Using Deep Learning" Technique, Turkish Journal, Volume 12, Issue 10, April 2021, p.3880-3892.

[4] Lizhen Tang, Qusay H. Mahmoud, "A Deep Learning-Based Framework for Phishing Website Detection", IEEE, Volume 10, December 2021, p. 1509 – 1521.

[5] Rundong Yang, KangfengZheng, Bin Wu, Chunhua Wu and Xiujuan Wang, "Phishing Website Detection Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning, Sensors" (Basel),doi: 10.3390/s21248281, December 202.

[6] Tristan Bilot, Gregoire Geis and Badis Hammi, "PhishGNN: A Phishing Website Detection Framework using Graph Neural Networks", SECRYPT At Lisbo, DOI:10.5220/0011328600003283, July 2022.

[7] Aman Rangpur, Tarun Kanakam and Dhanvanthini P, "Phish-Defence, Phishing Detection Using Deep Recurrent Neural Networks", Cornell University, Volume 4, September 2022 .

[8] Chenguang Wang, Yuanyuan Chen, "TCURL, Exploring hybrid transformer and convolutional neural network on phishing URL detection, Knowledge-Based Systems", Volume 258, Issue C, December 2022.

[9] Zainab Alshingiti ,Rabeah Alaqel ,Jalal Al-Muhtadi ,Qazi Emad Ul Haq,Kashif Saleem and Muhammad Hamza Faheem, "A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN", Vol 23,Jan 2023.

[10] ShouqAlnemari, Majid Alshammari, "Detecting Phishing Domains Using Machine Learning", Applied Sciences (2076-3417), Vol. 13, Issue 8, April 2023, p4649. 16p.

[11] Eman Abdullah Aldakheel, Mohammed Zakariah, Ghada Abdalaziz Gashgari, Fahdah A. Almarshad and Abdullah I. A. Alzahrani, "A Deep Learning-Based Innovative Technique for Phishing Detection in Modern Security with Uniform Resource Locators, Sensors", Vol. 23, Issue 9, April 2023.

[12] Manoj Kumar Prabakaran, Parvathy Meenakshi Sundaram ,Abinaya Devi Chandrasekar, "An enhanced deep learning-based phishing detection mechanism to

effectively identify malicious URLs using variational autoencoder, IET Information Security", Volume 17, Issue 3, May 2023, p. 423 – 440.

[13] Wei Wei, qiao ke Jakub Nowak, Marcin korytkowski, Rafat Scherer, Marcin wonxniak, "Accurate and fast URL phishing detector", Elsevier, Volume 198, September 2020.

[14] Mohamed A. El-Rashidy, "A Smart Model for Web Phishing Detection Based on New Proposed Feature Selection Technique", Menoufia J. of Electronic Engineering Research (MJEER), Vol. 30, No. 1, Jan.2023.

[15] Ahmet Selman Bozkir, Firat Coskun Dalgic, Murat Aydo, GramBeddings: "A New Neural Network for URL Based Identification of Phishing Web Pages Through N-gram Embeddings", Elsvier, Volume 124, January 2023.

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Literature Review on Phishing Website Detection Using Deep Learning

Akshatha A P[1], Chaithra R[1], Madhura S[1] , Chandana C Sagar[1] , Hiriyanna G S[2]

** 1UG students, Dept. of CS&E , JNNCE, Shivamogga, India.*
** 2Asst. Prof ., Dept. of CS&E , JNNCE, Shivamogga, India.*

Abstract - With the escalating threat of phishing attacks on the internet, the need for effective and efficient methods to identify phishing websites has become paramount. This study explores the application of Deep Learning (DL) techniques for the automated detection of phishing websites. Leveraging the power of neural networks, we analyze various features such as URL structure, content, and visual elements to develop a robust model. The proposed deep learning model exhibits high accuracy in distinguishing between legitimate and phishing websites, demonstrating its capability to adapt to evolving phishing techniques. The training process involves a diverse dataset of labeled examples. Experimental results on benchmark datasets showcase the model's superiority over traditional methods, marking a significant stride in the ongoing battle against cyber threats. The study contributes to the development of proactive measures to safeguard users against the pervasive and evolving nature of phishing attacks in the digital landscape.

Key Words: Deep Learning, Phishing detection, CNN, Neural Network.

1.INTRODUCTION

Phishing is a type of cybercrime involving technological and social approaches to collect financial and personal data from clients. Such as personally identifiable information, banking and credit card details, and passwords. Hacker obtains user information through various means, including email, forum postings, URLs, instant chats, text messages, and phone calls. Other than email and website phishing, there is also 'vishing' (voice phishing), 'smishing' (SMS Phishing) and several other phishing techniques cybercriminals are constantly arriving. With the rapid development of machine learning, there are more and more applications in the field of cybersecurity and we have proposed a deep learning-based framework to detect phishing links in a real-time web browsing environment. When the URL of the current tab of the browser is predicted to be a phishing link, the current page will receive an obvious warning prompt. The prediction result is obtained by the core prediction service calling a trained model.

2. LITERATURE SURVEY

Phishing websites continue to pose significant threats to online security, exploiting users' trust to steal sensitive information such as passwords and financial data. In response, researchers have conducted investigations into the detection methods and strategies to mitigate the proliferation of these malicious websites. This literature survey aims to synthesize and analyze the advancements in phishing website detection,

In [1] Link Calculator anti-phishing scheme is based on an algorithm designed to extract link characteristics from loading URLs to determine their legitimacy. Unlike the other link-based extraction approaches, the proposed approach introduced the concept of the weighting of the incoming request for its prediction without using the machine learning approach. The weighting concept allows the system to prevent superfluous computations on non-essential links within the parsed page. The advantage of this is to reduce the problems of false-positive and negatives occasioned by other methods where this idea is missing. This is because certain link information within parsed webpages or requests is sufficient to classify them as phishing without loss of generality.

In [2] the aim is to detect malicious URLs using minimum features by applying deep machine learning techniques. As an input web-page URLs are fed into the feature extractor. The feature extractor extracts the requisite features from the sources such as from URL, hyperlink and third party based and transfers them to Information Gain (IG) feature ranking algorithm. The IG algorithm supports in choosing the best performance features. The finest performance features are again trained over Deep Neural Network (DNN) to find out the output status and to differentiate between legitimate and phishing URLs. Here, a robust system based on deep learning neural network (DNN) is proposed which is highly efficient in detecting phishing websites. To train the deep learning model, URL heuristics and third party-based features have been used. Here we have minimized the number of features as compared to Rao and Pais, thereby reducing the dependence on third party-based amenities which is able to attain an accuracy of 99.90%.

In [3] paper they propose the combination of a convolution operation to model the character-level URL features and a deep convolutional autoencoder (CAE) to consider the nature of zero-day attacks. Extensive experiments on three real-world datasets consisting of 222,541 URLs showed the highest performance among the latest deep-learning methods. They demonstrated the superiority of the proposed method by receiver-operating characteristic (ROC) curve analysis in addition to 10-fold cross-validation and confirmed that the sensitivity improved by 3.98% compared to the latest deep model. The main innovation of this study is the introduction of deep anomaly detection to the field of phishing URL detection and achieving the best performance compared to classification-based deep-learning methods by implementing a neural network structure and an operation optimized for URL modelling. The combination of the encoding/decoding structure to facilitate disentanglement between classes and convolution operation optimized for character-level URL characteristics was utilized to define an anomaly score based on the reconstruction error.

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Anti-Phishing: A Web Identifier for Spoofed Sites Using Neural Network

Akshatha A P[1], Chaithra R[1], Madhura S[1], Chandana C Sagar[1] , Hiriyanna G S[2]

* *1UG students, Dept. of CS&E , JNNCE, Shivamogga, India.*
* *2Asst. Prof ., Dept. of CS&E , JNNCE, Shivamogga, India.*

Abstract -***Phishing attacks persist as a significant threat to cybersecurity, exploiting deceptive websites to illicitly acquire sensitive user information. Conventional anti-phishing techniques often struggle to keep pace with the evolving sophistication of these attacks. An approach for detecting phishing websites using Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) known for its ability to capture sequential dependencies in data.Method leverages the temporal dynamics inherent in website content and user interactions to discern between legitimate and phishing websites and constructed a comprehensive data-set comprising diverse samples of legitimate and phishing websites, ensuring the model's ability to generalize across various attack vectors.LSTM-based model exhibits resilience to adversarial evasion techniques commonly employed by attackers, demonstrating robustness in real-world scenarios and conducted extensive analysis to interpret the learned representations captured by the LSTM network, providing insights into the distinguishing features of phishing websites.***

Key Words:Phishing, Deep Learning,

1. INTRODUCTION

Phishing is a form of cyber attack where malicious actors attempt to trick individuals into disclosing sensitive information, such as login credentials, financial details, or personal data. The term "phishing" is a play on the word "fishing," as it involves luring victims with bait. These attacks typically take the form of deceptive emails, text messages, or websites that appear to be from legitimate sources, such as banks, social media platforms, or government agencies.

Phishing attacks often exploit psychological tactics, such as urgency or fear, to prompt victims to act quickly without questioning the authenticity of the communication.For instance, a phishing link might mimic a bank's website URL but feature a subtle alteration, redirecting unsuspecting users to a fraudulent site designed to harvest their login credentials. Unwary users who fall for these tricks may inadvertently provide their sensitive information to cybercriminals.

There are several common types of phishing attacks, including:

1. Email Phishing:This is the most prevalent form of phishing, where attackers send deceptive emails purporting to be from trusted organizations. These emails often contain links to fake websites or malicious attachments designed to steal information.

2. Spear Phishing: In spear phishing attacks, cybercriminals target specific individuals or organizations, often using personalized information to increase the likelihood of success. This could involve using the victim's name, job title, or other details obtained through research.

3. Whaling:Whaling attacks target high-profile individuals, such as executives or celebrities, with the aim of stealing sensitive information or perpetrating financial fraud. These attacks often involve sophisticated tactics and social engineering techniques.

4. Smishing:Smishing, or SMS phishing, involves sending fraudulent text messages to trick recipients into revealing personal information or clicking on malicious links. These messages often claim to be from banks, delivery services, or other trusted organizations.

5. Vishing:Vishing, or voice phishing, involves using phone calls to deceive victims into providing sensitive information or performing certain actions. Attackers may impersonate legitimate organizations or individuals to gain the victim's trust.

Phishing websites are fraudulent online platforms designed to imitate legitimate websites, often with the aim of deceiving users into divulging sensitive information such as passwords or financial details. These malicious sites typically employ various tactics to appear authentic, including mimicking the visual design and branding of reputable organizations or using deceptive URLs. Once users enter their information, cybercriminals exploit it for nefarious purposes, such as identity theft, financial fraud, or unauthorized access to accounts.

With the rapid development of machine learning, there are more and more applications in the field of cybersecurity, and we have proposed a deep learning-based framework to detect phishing links in a real-time web browsing environment. When the URL of the current tab of the browser is predicted to be a phishing link, the current page will receive an obvious warning prompt. The prediction result is obtained by the core prediction service calling a trained model.

2. IMPORTANCE OF PHISHING WEBSITE DETECTION

Detecting phishing websites is paramount in safeguarding personal and financial security in today's digital landscape. These deceptive websites are crafted to mimic legitimate platforms, aiming to trick unsuspecting users into divulging sensitive information like passwords, credit card details, and personal identifiers. By identifying and blocking phishing sites, individuals and organizations can prevent dire consequences such as identity theft, financial loss, and reputational damage. Moreover, swift detection helps curb the dissemination of malware, which often accompanies phishing