International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET

Intelligent Rule-Based Phishing Websites and Malicious URL Classification Based on URL Features

Shanthi D[1], Hemalatha S[2], Karthikeyan R[2], Murugeshwari N[2], Smitha K Varghese[2]

[1]Assistant Professor, Department of Computer Engineering, Coimbatore Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India

[2]UG Scholar, Department of Computer Engineering, Coimbatore Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India

Article Info
Volume 8 Issue 2
Page Number: 156-165

Publication Issue :
March-April-2021

Article History
Accepted : 10 March 2021
Published : 15 March 2021

ABSTRACT

Malicious URLs are one of the biggest threats to this digital world and preventing it is one of the challenging tasks in the domain of cyber security. Previous research to tackle malicious URLs using hard-coded features have proven good indeed, but it comes with the limitation that these features are non-exhaustive and therefore detection algorithms fail to recognize new or unseen malicious URLs. However, with the deep learning revolution, this problem can be easily solved, since deep learning models extract features of their own by learning from patterns occurring in such URLs.In this project, we are proposing rnn and cnn based algorithm which can be effective for classifying URLs as malicious or benign. With the continuous development of Web attacks, many web applications have been suffering from various forms of security threats and cloud computing attacks.

Keywords : URL, Phishing, CNN, DNN, RNN

I. INTRODUCTION

The security detection of URLs has always been the focus of Web security. Many web application resources can be accessed by simply entering an URL or clicking a link in the browser. An attacker can construct various web attacks such as SQL, XSS, and information disclosure by embedding executable code or injecting malicious code into the URL. Therefore, it is necessary to improve the reliability and security of web applications by accurately detecting malicious URLs. This project designs a recurrent neural network and CNN for the detection of malicious URLs detection based on characters as text classification features. Considering that malicious keywords are unique to URLs, a feature representation method of URLs based on malicious keywords is proposed, and a GRU is used in place of the original pooling layer to perform feature acquisition on the time dimension, resulting in high-accuracy multicategory results.

II. PROPOSED SYSTEM

Cyber-attacks have become an international threat worldwide - confidential and secret data is being compromised which can harm national security. Two domains of the IT industry are majorly compromised which primarily consists of information processing and physical infrastructure supporting the first domain. Phishing is one of the most popular attacks that malicious groups carry out towards the general users. Phishing is the vindictive attempts to gain sensitive information like username, passwords, credit card information etc. which is often disguised as trustworthy source to the innocent users. When combined with Social engineering which often differs for different users, they may find very tempting to log in their credentials. This makes phishing one of the most dangerous and malignant attack. Although Phishing websites are very common, there are numerous methods to detect are prevent these attacks. These range from URL detection where a phishing website can be detected using some of the key features based on their Uniform Resource Locator to the use of Machine Learning Algorithms like Random Forest, Artificial Neural Network or Select Vector Method so that to stop these websites dynamically. There are also some methods that use Blacklist approach, where the system refers to anti-phishing repositories like Phish tank or Yahoo Phishing database to check whether the site is a phishing website or a legitimate one. This review paper explains many different methods used for the detection and prevention of phishing websites explaining the above-mentioned techniques in muchmore detail and also many other methods that help the users stay safe and secure from this type vicious attack.Phishing attack happens when a malicious website mimics a legitimate and trustworthy website to trick the users into submission of their sensitive credentials. The main motive of these malignant websites is to harvest the credentials of different users and to use this for various fraudulent activities. Nowadays users can perform various online activities like sending/receiving emails, banking transactions, buy/sell goods and others. Such activities are susceptible to phishing attacks and many victims suffer due to it. The victims may lose information in this process. Hence, due to the ignorance of internet users, the phishers exploits the security. Two types of issues occur with phishing - technical and non-technical (human). To obtain a powerful solution, both issues should be handled effectively. Over a time, multiple software has been designed to tackle the overlooked situation of phishing. Techniques for phishing can be classified as web content, industrial toolbar and user interface based anti-phishing. These methods cover filtering, authentication, analyzing and attack tracking. These services, however, all phishing attacks are not blocked/stopped by them. Latest studies show that the use of Machine Learning can help tackle this type of attack. Machine Learning is a branch in Artificial Intelligence that that gives the system the functionality of learn things automatically without someone explicitly programming it. The use of algorithms like Artificial Neural Network, Naive Bayes can be used for the detection and prevention of Phishing websites. Artificial Neural Network is a archetype of Machine Learning algorithm where information communication is inspired by the actual working of neurons. It helps the system to have adaptive learning and self-organization. Its applications range from self-driving automatic vehicles to fraud detection. Naive Bayes is a type of Machine Learning algorithm where classification is done with the help of our understanding of probability. The real-world applications of Naive Bayes range from email spam detection to facial recognition and much more. These and many other algorithms can be used for detection and prevention of phishing websites. In this survey paper, many machine learning algorithms are compared with one another to analyse their accuracy over different standard metrics on a common data-set. According to

APWG survey report the absolute of phish identified in 1Q2018 was 263,538. This was raised up 46 percent from the 180,577 examined in 4Q 2017 as shown in Fig. 1. It was also compellingly more than the 190,942 seen in 3Q 2017.

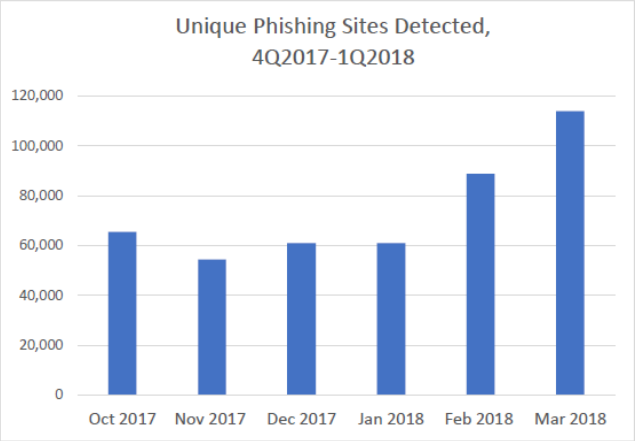


Fig. 1. Total number of submitted unique

In this modern world everyone is accessing the internet for different purpose. As the internet grew; there are some problems like stealing data, hacking and fake web pages came into being. To get rid of these phishing websites 'phishing website detection website' is very useful. It helps users to understand what phishing is and how to stay safe while surfing the internet, because people use their personal information like credentials and bank account details. This project has three main parts. In the first part it has 'URL detection' method. In this part what it does is when we access to the webpage, we don't know that the web page we are accessing is safe or not. So, in this first part when you copy the URL and search it tells you weather the URL is phishing or not. In the second part 'Mail phishing' it tells user to copy the email header and paste in to the given header to analyze it. Then it displays the result. In the last part called 'Wiki' it tells you about the basic types of phishing and gives the basic idea of what is phishing for the new user who doesn't have any prior knowledge. So, it is very useful nowadays because we have to do most of the things on the internet, so it should be safe and this project achieve this goal.

Basic steps for phishing:

- Planning: attacker decides which business or organization to target and how to get email ids of the customers for that organization. They often use the mass-mailing technique.
- Setup: Once they know which business to spoof and what their customers are, they find a way to deliver the message with the spoof links or documents with company's logo to get trust from customers.
- Attack: This is the most familiar step for everyone; in this step phisher sends a phony message that seems like from reputable source.
- Fraud: The data or information phisher gathered; they use it to fraud like transfer money from other account or make illegal purchase.

Scope

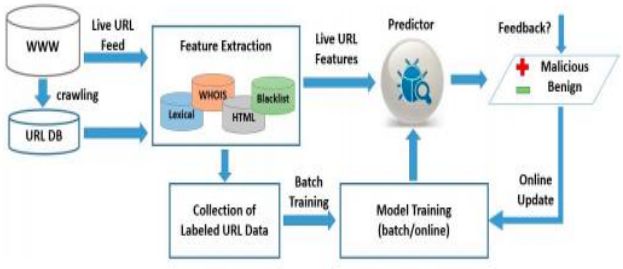
Phishing is a considerable problem which differs from the other security threats such as intrusions and Malware which are based on the technical security holes of the network systems. The weakness point of any network system is its Users. Phishing attacks are targeting these users depending on the trikes of social engineering. There are many possible ways to do phishing but unfortunately, we have some techniques available nowadays can stop spear and email phishing and therefore we need to build a system which can help us to stop phishing attacks. One real example is the Google mail phishing attack happened some years ago and many users lost their trust on Google. So, to avoid all these big threats, we need to think possible ways to avoid large scale attacks to protect users. The scope for the phishing in this project is to save maximum users on the web and the one who work on emails. The motto of the PHISHING DETECTION project is when user is about to enter any information

such as personal info or bank information we recommend them to search that link or analyze the header of that email in the web detect site first and if it shows the result is legitimate then they are safe to give the information over the internet. The scopes of the project are as follows:

Phishing Detection • To develop a system which can cover the large scope of safe surfing on the internet with using URL detection method; URL is the most common type user interact with possible viruses and attacks

- To begin the awareness of current threat and possible future fraud methods for all users which can educate them about how to stay away from attacks and spam websites and not to become them bate
- A good practical solution for big companies to keep their customer safe and win their trust and give satisfied services.
- Mail header analyzing system introduced for big organizations and people as a good filter to avoid scam happened through emails and links which lead them to fake web page.
- A very quick and user-friendly system to check and proceed with safety to save time, money, energy and peace of mind.

III. METHODOLOGY



This section describes our proposed RNN model for malicious URL detection. This model combines the characteristics of URLs in the field of Web attacks at the character level. It uses gated recurrent units And CNN to extract features from URLs. Finally, it combines the classification module to detect Web attacks in URLs.

Our neural network model is divided into three parts:

- Keyword-Based URL Character Embedding
- Feature Extraction Module
- Classification Module

KEYWORD-BASED URL CHARACTER EMBEDDING:

The character embedding module is used to map the original URL character into a low-dimensional vector, thereby encoding the original sequence as a two-dimensional floating-point matrix. In the character embedding, the malicious keyword in the URL is distinguished from the ordinary character. Such differentiation can highlight the key part in the URL, which is advantageous in allowing the feature detection module to extract the representative feature more quickly.

IV. FEATURE EXTRACTION MODULE

The feature detection module uses the convolutional neural network to extract features on the abstract level of the URL and uses the GRU as a pooling layer, retaining the important features on the premise of preserving the context relationship. It uses a combination of different-length convolution windows to more fully extract features at each level. To make full use of the extracted features, the features extracted from the convolution windows need to be merged. The pooling section after the convolution operation uses the GRU as a pooling layer.

CLASSIFICATION MODULE

The fully connected neural network is used to classify the detected features. In our detection model using a model file created which finally tells the website is safe to use or not

Recurrent Neural Networks (RNN) and types

The RNN structure

Generally there are two kinds of neural networks which are feedforward neural networks and recurrent neural networks. A feedforward neural network is an artificial neural network where connections between the units do not form a cycle like Figure 1 In other word in feedforward networks processing of information is piped through the network from input layers to output layers.

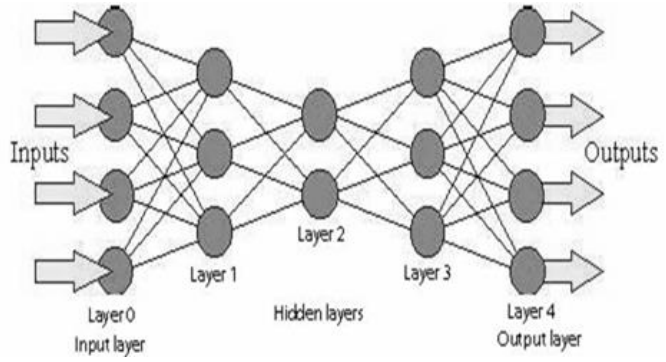


Figure 1. An example of feedforward neural network

In contrast, a recurrent neural network (RNN) is an artificial neural network where connections between units form cyclic paths. RNNs are called recurrent because they receive inputs, update the hidden states depended on the previous computations, and make predictions for every element of a sequence. By unrolling an RNN in time, it can be considered as a deep neural network (DNN) with indefinitely many layers (Figure 2).

which is required to initialize the first hidden state is typically set to all zeroes. The output of the network is y which is calculated by a nonlinear function of matrix multiplication of V and S_t . In fact this nonlinear function, g, is the activation function for the output layer and usually it is the softmax function. It is simply a way to convert raw scores to probabilities. Unlike feed forward neural networks, which have different parameters at each layer, a RNN shares the same parameters (U, V, W in Eq. 2.1 and Eq. 2.2) across all steps.

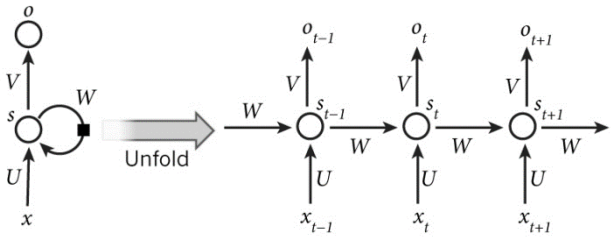


Figure 2. An RNN unrolled in time

We can consider RNNs as neural networks with memory to keep information of what has been processed so far. RNNs are very powerful dynamic systems for sequence tasks, such as speech recognition or handwritten recognition. They are powerful because they can maintain a state vector that implicitly contains information about the history of all the past elements of the sequence [4].

The RNN depicted in **Error! Unknown switch argument.** network makes predictions by matrix multiplications as follows:

$$S_t = f(Ux_t + Ws_{t-1}) \quad (2.1)$$

$$y = g(Vs_t) \quad (2.2)$$

In these equations x_t is the input at time step t. S_t is the hidden state at time step t which is in fact the memory of the network and it is calculated based on the input at the current step and the previous hidden state. f is a activation function which transforms the inputs of the layer into its outputs and allows us to fit nonlinear hypotheses. Common choices for f are tanh and ReLUs. S_{-1} which is required to initialize the first hidden state is typically set to all zeroes. The output of the network is y which is calculated by a nonlinear function of matrix multiplication of V and S_t. In fact this nonlinear function, g, is the activation function for the output layer and usually it is the softmax function. It is simply a way to convert raw scores to probabilities. Unlike feed forward neural networks, which have different parameters at each layer, a RNN

shares the same parameters (U, V, W in Eq. 2.1 and Eq. 2.2) across all steps.

RNN training

There are different approaches for training of RNNs including back-propagation through time (BPTT), real-time recurrent learning (RTRL), and extended Kalman filtering approaches (EKF). In this study we mostly focus on BPTT.

The feedforward neural networks can be trained by backpropagation algorithm. In RNNs, a slightly modified version of this algorithm called Backpropagation through Time (BPTT) is used to train the network. The backpropagation algorithm can be extended to BPTT by unfolding RNN in time and stacking identical copies of the RNN. As the parameters that are supposed to be learnt (U, V and W) are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps.

In RNNs, a common choice for the loss function is the cross-entropy loss which is given by:

$$L(y_l, y) = -\frac{1}{N}\sum_{n\in N} y_{ln} \log y_n \qquad (2.3)$$

In this formula, y_l is the number of training examples, y is the prediction of the network and y_l is true label. The parameters U, V and W can be calculated during training by minimizing the total loss on the training data. One popular approach to do this is Stochastic Gradient Descent (SGD). The idea behind SGD is iterate over all our training examples and during each iteration, we update the parameters into a direction that reduces the error. These directions are calculated by the gradients on the loss function respect to U, V and W: $\frac{\partial L}{\partial U}, \frac{\partial L}{\partial V}$ and $\frac{\partial L}{\partial W}$. In fact BPTT can be considered as a black box that gets training data as input and returns these gradients.

The Vanishing Gradient Problem

While RNNs are powerful structures, practically they are hard to train. One of the main reason is "vanishing gradient problem" which explored in depth by Bengio[5][6]. They found that in theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps. This means in practice the range of contextual information that standard RNNs can access are limited. The vanishing gradient problem is illustrated schematically in **Error! Unknown switch argument.**. It is proved that the influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially as it pipes through RNN. In fact, it is hard for an RNN to bridge gaps of more than about 10 time steps between relevant input and target events [7].

Fortunately there are a few approaches to overcome this shortcoming of RNN. For example, W matrix can be initialized properly to combat the vanishing gradient problem or using ReLU instead of tanh or sigmoid activation functionscan reduces the effect of vanishing gradients. However, the most successful solution is to use Long Short-Term Memory (LSTM) which will be introduced in next chapter.

Deep Recurrent Neural Networks (DRNN)

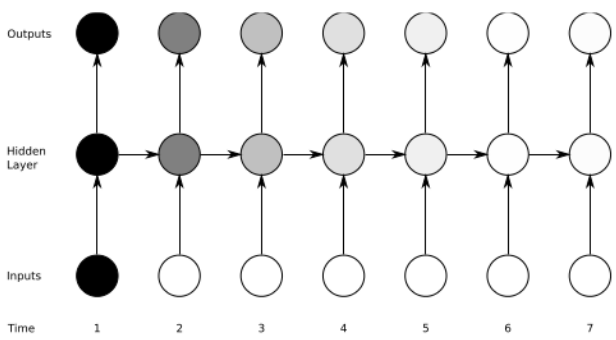


Figure 3. Vanishing gradient problem in RNNs

DRNN structure

One potential disadvantage of traditional RNN which explained in part 1 is that, the information only passes through one layer of processing before going to the output. In sequence tasks we usually need to process the information at several time scales. In this part a structure called deep recurrent neural network (DRNN) will be explained which is basically a combination of the concepts of deep neural networks (DNN) with RNNs [1]. In DRNN by stacking RNNs, every layer is an RNN in the hierarchy that receives the hidden state of the previous layer as input. In this way different time scales at different levels, and therefore a temporal hierarchy will be created.

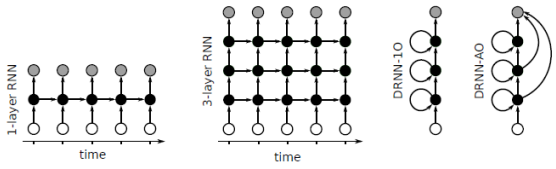


Figure **4**. Different variations of the structures of DRNNs

Different variations of the structures of DRNNs are depicted in Figure . In this picture connection matrices are shown by arrows. Also input frames, hidden states, and output frames are represented by white, black and grey circles respectively. In left a traditional RNN unrolled in time is shown. In middle we can see a DRNN with three layers folded out in time. In right two variants of DRNN structures are shown. In DRNN-1O the output will be computed just based on the hidden state of the last layer, while in DRNN-AO the output will be the combination of the hidden states of the all layers. In both of these structures the looped arrows represent the recurrent weights.

DRNN-AO has two great advantageous in comparison with DRNN-1O. Firstly, when we are using BPTT for DRNN-AO, the error will propagate from the top layer down the hierarchy without attenuating of magnitude and as a result the system will be trained effectively. Secondly, it gives us the ability to evaluate the impact of each layer in solving the task by leaving out an individual layer's contribution. In this part character-based language modelling will be explored using both of DRNN-1O and DRNN-AO.

As the distribution of characters in this task is quit nonlinear and covers both short and long term dependencies, it is a well-suited task to study temporal hierarchy of DRNNs. Here the experiments ran on Wikipedia-based text corpus using only stochastic gradient descent (SGD).

We assume that our DRNN has L layers and N neurons per layer and input is a time series x(t) with dimensionality of N_{in} and y(t) is the output of DRNN. Then we have:

$$s_t^l = \tanh\left(U^l x_t + W^l s_{t-1}^l\right) \quad \text{if } l = 1 \tag{4.1}$$

$$s_t^l = \tanh\left(U^l s_t^{l-1} + W^l s_{t-1}^l\right) \quad \text{if } l > 1 \tag{4.2}$$

As it is mentioned before, two different structures for generating outputs are candidate. The output for DRNN-1O is:

$$y_t = \text{softmax}\left(V s_t^L\right) \tag{4.3}$$

And the output for DRNN-AO is:

$$y_t = \text{softmax}\left(\sum_{l=1}^{L} V^L s_t^l\right) \tag{4.4}$$

Note that how these equation are similar to Eq 2.1 and Eq 2.2.

DRNN training

In order to train a DRNN, BPTT approach can be applied using stochastic gradient decent for optimizing the parameters as follows:

$$\theta(j+1) = \theta(j) - \eta_0 \left(1 - \frac{j}{T}\right) \frac{\nabla_\theta(j)}{\|\nabla_\theta(j)\|} \tag{4.5}$$

In Eq. 4.5, $\theta(j)$ is the set of all trainable parameters after j updates, and $\nabla_\theta(j)$ is the gradient of a cost function with respect to this parameter set, as computed on a randomly sampled part of the training set. T is the number of batches and the learning rate is set at an initial value η_0 which decreases linearly with each subsequent parameter update.

For DRNN-1O, an incremental layer-wise method should be used which means training the full network with BPTT and linearly reducing the learning rate to zero before a new layer is added. Then the layers will be added one by one, and after adding a layer the previous output weights will be discarded, and new output weights are initialized connecting from the new top layer. For DRNN-AO, we can test the influence of each layer by setting it to zero. This can assure us that model is efficiently trained.

Performance of DRNNs

Using the methods explained in 4.2., two DRNNs including both of DRNN-1O and DRNN-AO is trained for Wikipedia character prediction task. The results of these experiments are shown in table 4.1.

Model	BPC test
RNN	1.610
DRNN-AO	1.557
DRNN-1O	1.541
MRNN	1.55
PAQ	1.51

Table 4.1. Results of DRNN on Wikipedia Corpus [1]

These results show that DRNN can improve the performance of the network for the task of language modeling significantly and in fact it can gives us a state of the art performance using SGD. It is proved that DRNN have a hierarchy of time-scales in their layer. Additionally, it is demonstrated that in certain cases the DRNNs can have extensive memory of several hundred characters long.

We are going to use following algorithms to classify the websites.

1) RNNalgorithm
2) Naïve Bayesian Algorithm

1) RNN algorithm-

RNN are a powerful prediction method used to predict based on the training dataset. The representation of the CART model is a binary tree. A node represents a single input variable (X) and a split point on that variable, assuming the variable is numeric. The leaf nodes (also called terminal nodes) of the tree contain an output variable (y) which is used to make a prediction.

Once created, a tree can be navigated with a new row of data following each branch with the splits until a final prediction is made. Creating a binary RNN is actually a process of dividing up the input space. A greedy approach is used to divide the space called recursive binary splitting. This is a numerical procedure where all the values are lined up and different split points are tried and tested using a cost function. The split with the best cost is selected. All input variables and all possible split points are evaluated and chosen in a greedy manner based on the cost function.

The Gini index is the name of the cost function used to evaluate splits in the dataset. A split in the dataset involves one input attribute and one value for that attribute. It can be used to divide training patterns into two groups of rows. Once the best split is found, we can use it as a node in our decision tree. We need to decide when to stop growing a tree. We can do that using the depth and the number of rows that the node is responsible for in the training dataset.

Maximum Tree Depth-This is the maximum number of nodes from the root node of the tree.
Once a maximum depth of the tree is met, we must stop splitting adding new nodes.

Minimum Node Records- This is the minimum number of training patterns that a given node is responsible for. Once at or below this minimum, we must stop splitting and adding new nodes.

Making predictions with a RNNinvolves navigating the tree with the specifically provided row of data.

The last node of the tree is known as terminal node, and it is responsible for the prediction.

Classification metrics which are considered to evaluate a classifier are:

1) Accuracy - Accuracy rate is the percentage of test set samples that are correctly classified by the model.
2) Precision – exactness- what % of tuples that the classifier labeled as positive are actually positive.
3) Recall - completeness – what % of positive tuples did the classifier label as positive.
4) F-score - harmonic mean of precision and recall.

Purpose

Phishing attack is used to steal confidential information of a user. Fraud websites appears like genuine websites with the logo and graphics of genuine website. This project aims to detect fraud or phishing website using data mining techniques. W3C standard defines characteristics which can be used to distinguish a legal and fraud website. These characteristics can be used for classification. By using data mining techniques, we will generate a classification model which is used to manage and model data. This helps to make a prediction whether the website is legal or fraud.

1. Expected outcomes and result:

We will use a data mining techniques that will generate a classification model from a training dataset and then this model will be applied to testing dataset which will label the websites as malicious or legitimate. We will compare the accuracy of two models generated and conclude which is better among those two.

2. Intermediate steps of project:

1) Collect dataset containing malicious and legitimate websites.
2) Writing code to divide the dataset into training and testing sets and code for displaying the evaluation results.
3) Run both the classification algorithms on the dataset.
4) Compare the results for the two models generated using two algorithms and specify which is better.

V. CONCLUSION

From this project, it can be inferred that there are many possible ways to detect and prevent phishing attacks. We proved Machine learning methods are best to analyse URLs to identify their potential peculiar characteristics. While the other method checks the web server log files to identify whether the phishing website did some transaction of resources with the legitimate website. Hence tracking it down and blocking it. Machine Learning plays a very vital role in this field. It enables a dynamic approach for detecting and preventing phishing websites.

VI. REFERENCES

[1]. Jun Ho Hun and Hyoungshick Kim, “Phishing Detection with Popular Search Engines: Simple and Effective ,” in, Springer-Verlag Berlin Heidelberg, 2012.
[2]. Choon Lin Tan, Kang Leng Chiew, San Nah Sze, “Phishing Website Detection Using URL-Assisted Brand Name Weighting System ,” in, IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS) December 1-4 , 2014.
[3]. Phoebe Barraclough, Graham Sexton, “Phishing Website Detection Fuzzy System Modeling ,” in, Science and Information Conference 2015 July 28-30, 2015.
[4]. Giovanni Armano, Samuel Marchal, N. Asokan, “Real-Time Client-Side Phishing Prevention

Add-on," in, IEEE 36th International Conference on Distributed Computing Systems, 2016.

[5]. Abdulghani Ali Ahmed,Nurul Amirah Abdullah, "Real Time Detection of Phishing Websites," in, IEEE 2016.

[6]. Huu Hieu Nguyen and Duc Thai Nguyen, "Machine learning Based Phishing Websites Detection," in, Springer International Publishing Switzerland 2016.

[7]. Jun Hun, Xiangzhu Zhang, Yuchun Ji, Hanbing Yan, Li Ding, Jia Li and Huiming Meng, "Detecting Phishing Websites Based on Study of the Financial Industry Webserver Logs ," in, 3rd International Conference on Information Science and Control 2016.

[8]. Varsharani Ramdas Hawanna, V. Y. Kulkarni, "A Novel Algorithm to Detect Phishing URLs," in, International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT) International Institute of Information Technology(I2IT), Pune, 2016.

Cite this article as :

Shanthi D, Hemalatha S, Karthikeyan R, Murugeshwari N, Smitha K Varghese, "Intelligent Rule-Based Phishing Websites and Malicious URL Classification Based on URL Features ", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 8 Issue 2, pp. 156-165, March-April 2021.
Journal URL : https://ijsrset.com/IJSRSET218246