Detection of Phishing Websites by Investigating Their URLs using LSTM Algorithm

Barah Mohammed Alanzi[1] and Diaa Mohammed Uliyan[1]
Bh.alghassab@gmail.com, d.uliyan@uoh.edu.sa
[1]College of Computer Science and Engineering,
Department of information and computer science
University of Hai'l ,Ha'il, Saudi Arabia

Abstract
Phishing is a criminal mechanism that uses both social engineering and technical tricks to steal consumers' personal identity data and financial account credentials. As the number of web user's increases, phishing frauds are gradually increasing. In order to respond effectively to various phishing mechanism, a proper understanding of phishing attacks is necessary, and some appropriate response methods should be utilized. In this paper, the main aim is to detect a phishing website attack by a suggested machine learning algorithm. First, we need to update a blacklisted URLs and IP for antivirus into the database of our method. The database is known as the "blacklist". Second, to avoid blacklist attackers, we need to understand how they use creative techniques to deceive users by modifying the URL to look as legitimate user via obfuscation and many other simple techniques including fast blur, where proxies are automatically generated to host a web page; Algorithm generation of new URLs; etc. A blacklist is a list of many unsafe websites that are accused of fraud, spreading malware, or launching any other form of malicious activity. Having this list is one of the biggest nightmares for website owners because the websites that became part of this list are no longer scanned by web crawlers, and there are no backlinks to create these sites. The first step is to collect benign and phishing URLs. Then, Host-based, popularity-based, and lexical feature extractions are applied to create a database of feature values. Finally, the database is knowledge extracted using various methods of machine learning. An experimental study was conducted using a deep learning algorithm, including long-term memory (LSTM). To analyse the behaviours of these deep learning architectures, extensive experiments were conducted to examine the effect of parameter tuning on the performance accuracy of deep learning models. The experimental results from this paper also show several issues and suggest future research directions related to deep learning in the field of phishing detection.

Keywords:
URL; Phishing website; Machine learning; LSTM; RNN.

1. Introduction

With the increase in the number of web users, phishing attacks are gradually increasing. In order to respond effectively to various phishing attacks, a proper understanding of phishing attacks is necessary, and appropriate response methods should be used.

Manuscript received May 5, 2022
Manuscript revised May 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.5.60

Online URL reputation services are used to classify URLs and the classes returned are used as a supplementary source of information that will enable the system to classify URLs. The classifier achieves an accuracy of 94-96% by detecting a large number of phishing hosts, while maintaining a modest false positive rate. URL groups, URL categorization, and URL grading mechanisms work in tandem to give URLs a rank [8].

Using Long-Short-Term Memory (LSTM) provides an effective solution for detecting phishing sites. The LSTM algorithm can solve more complex problems compared to shallow learning algorithms (that is, traditional machine learning algorithms). Moreover, LSTM can store past information for a long time, however, Recurrent Neural Networks (RNN) are not able to do this task for long periods. LSTMs have an internal state, they are familiar with the temporal structure of the inputs, and they can model the parallel chain of inputs separately. As such, we aimed to combine the power of the LSTM algorithm into a single model and presented how to implement this integration effectively [10].

1.1 Research problem

The researcher formulated the research questions according to the purpose of the study, which are as follows:

1) How does machine learning by detecting URLs identify phishing sites?

All machine learning models use features - properties or attributes of data extracted from input data sets to create their models. In the context of URL classification, there are three types of features: host-based, lexical, and content-based. Host-based features are those that identify identity, location, and other network information about the host. Lexical features are text properties that are obtained from the URL itself. Finally, content-based features come from web pages linked to the same URLs. Content-based features require more in-depth analysis of the content and are computationally more expensive. It is also an inherent risk that our systems could be compromised while exploring web pages related to the URLs we are trying to rank for. The set of content-based features is outside the scope of our research, due to the associated risks and greater time requirements.

2) How to apply ML methods to classify malicious and legitimate websites?

The researchers examined a variety of techniques to prevent phishing attacks. Anti-phishing technologies can be categorized into three categories: email-based, content-based, and URL-based detection methods. The researchers used machine learning techniques on feature sets derived from phishing emails. Other studies in classifiers using features of URLs and search engine results as a means have investigated high detection rates while maintaining low false positive rates. Chandrasekaran et al. Attempting to identify phishing emails using structural properties. These anti-phishing techniques are not always applicable because the methods require phishing emails [13].

The authors developed URL-Net, a CNN-based deep neural URL discovery network. They argued that current methods often use Bag of Words (BoW) properties but suffer from some fundamental limitations, such as a lack of detection of concatenated concepts in the URL string, a lack of automatic feature extraction and a failure of properties that are not actually visible in ephemeral URLs.

The authors developed CNNs and Word CNNs to form the visual image of the network. In addition, they suggested modern techniques that were effective in dealing with rare terms, a prevalent issue in malicious URL detection tasks. This approach can allow URL-Net to identify embedding's and use sub-word data from invisible words during the testing phase.

The main problem addressed in this study is to enhance user authentication on a website. The research investigates the potential uses of the input models in detecting phishing URLs. In particular, the goal here is to develop the model that will be used to predict whether a website is fraudulent or legitimate and, if so, to what degree, to improve the accuracy of phishing URL detection [11].

1.2 Research objectives

The goal of a phishing site is to obtain personal information without permission, either through extortion or by visiting a fake web page that looks like the real one, which asks the user to enter personal information. This results in information security breaches through compromises in confidential data where the victim may suffer financial loss or loss of assets. The attacker may additionally commit identity theft using the personal details of the victims. Also, a phishing attack can damage the reputation of the spoofed financial institution, as customers lose confidence that their account is secure. Thus, they may take their habits to another company. Phishing, if not investigated, can negatively affect an organization's assets, revenue, customer relationships, or marketing efforts, as well as a company's image. A phishing attack could cost the company hundreds of thousands of dollars per attack in terms of employee time and fraud-related loss [9].

The main objective of this paper is to analyse the performance of the LSTM algorithm in detecting phishing activities. This analysis will help organizations or individuals choose and adopt the appropriate solution according to their technology needs and specific application requirements to combat phishing attacks [24].

1.3 Importance of this research

The harmful effects of phishing can be to gain access to user's confidential details, which can lead to financial losses to users and even prevent them from accessing their own accounts. Therefore, in this study, we will identify and qualify phishing website features to prevent and mitigate the risks of phishing website.

In addition, this study will make a comparative evaluation between the techniques of machine learning algorithms.

Online phishing costs internet users billions of dollars annually. Scammers steal personal information and financial account details such as usernames and passwords, leaving users vulnerable in the online space.

1.4 Limitations and delimitations of the study

Since the problem of phishing takes advantage of human ignorance or naivety regarding their interaction with electronic communication channels (such as email, HTTP, etc.), it is not always easy to solve. All suggested solutions try to reduce the impact of phishing attacks.

From a high-level perspective, there are generally two popular proposed solutions to mitigate phishing attacks:

• User education. Humans are taught to try to improve classification accuracy to correctly identify phishing messages, and then apply appropriate actions to properly categorized phishing messages, such as reporting attacks to system administrators.

• Software improvement. The program was developed to better categorize phishing messages on behalf of the user, or to present the information in a more simplified manner.

The main disadvantages of both approaches are:

• Resistance to training by non-technologists, so training must be permanent.

• Still, some software solutions depend on the user. If people ignore the security warnings, the solution may become useless [15].

Attackers can use technical vulnerabilities to create socially customized packets (such as using legitimate but deceptive domain names). Effective mitigation requires addressing matters on a personal and technical level. Since phishing attacks aim to exploit vulnerabilities of the user (i.e. end users), it is difficult to minimize them. For example, according to the assessment, end users fail to detect 27% of phishing attacks, even when taught with the best outreach software. On the other hand, software phishing detection techniques are evaluated against phishing attacks, which makes their performance abnormal by phishing tactics [16].

2. Literature review

The researchers examined a variety of techniques for preventing phishing attacks. Anti-phishing technologies can be classified into three categories: email-based, content-based, and URL-based detection methods. The researchers used machine learning techniques on feature sets derived from phishing emails.

Several researchers have analysed the statistics of suspicious URLs in some way. Our approach borrows important ideas from previous studies. We are reviewing previous work on phishing Site discovery using URL features that motivated our own approach.

Bahnsen et al. (2018) [1] suggested a more effective method for real-time phishing URL detection. It was mentioned that there are a lot of anti-phishing methods appearing, but scammers use diverse and dynamic methods for scam victims, so a smart and flexible model was needed to catch a phishing URL. Data mining methods can be used to promote an active model that contains basic and non-trivial data that can be backed up from huge data sets using classification algorithms to name a legitimate URL.

Four different classification algorithms were used to classify and approximate the data set for its achievement, accuracy, and several criteria. The experiments were handled using four different rule-based algorithms to detect cryptic awareness, from the huge data set to predict a phishing URL. The rated results paralleled their performance on the accuracy chart, error rate, time duration, and total number of component criteria. However, the results showed that all the selected algorithms complete a higher expected rate. The rules that were developed demonstrated the interaction and relationship between URL features which can help us build frameworks for detecting phishing URLs. There was a phishing detection form which is good for preventing users from being deceived by achieving verification by sending private information.

Preethi, and Velmayil (2019) [2] suggested a method for analysing phishing URLs using lexical analysis. Suggest a pre-phishing algorithm which is computerized machine learning to resolve phishing and non-phishing URLs to extract safe results.

Phishing URLs often contain two connections between the part of the registered domain level and the method or reservation level URL. Therefore, applying a URL to connections describes threading and categorizes using feature extractor from attributes. Also, these features are then used in a machine learning method to catch phishing URLs from an actual data set. Phishing and non-phishing URLs were categorized by detecting the domain value and the threshold value for each attribute using decision rating. This technique was further classified in Mat Lab using three major classifiers SVM, Random Forest and Naive Bayes to discover how it works in data set estimation. This paper suggested that the pre-phishing algorithm for an effective phishing URL detection system is based on the analysis of the URL sentence. The Pre-Phish approach was Demo Phishing, an empirical case study investigated to collect and evaluate a variety of phishing URL features and patterns, with all relevant attributes. This was a computerized machine learning method that relied on the characteristics of a phishing URL to catch and block phishing URLs and to provide a high level of security. The same limitations have been used to create a tool based on a web browser plug-in that can capture and block phishing URLs in real time and resolve data mining methods to detect new patterns of phishing URLs [2].

Ashit Kumar Dutta (2021) [3] proposed study emphasized the phishing technique, whereby a phishing URL is seen to include the automatic classification of URLs in a predefined set of category values based on several features and a category variable. Machine learning-based phishing techniques rely on URL functions to gather information that can help categorize URLs to detect phishing sites. To combat the ongoing complexity of phishing attacks and tactics, anti-phishing techniques are essential. The authors used LSTM technology to identify malicious and legitimate URLs. The crawler was developed that crawled 8000 URLs from the Alexa Rank portal, and also used the Phish-tank dataset to measure the efficiency of the proposed URL detector. The result of this study shows that the proposed method provides superior results instead of the current deep learning methods. A total of 8000 malicious URLs were detected using our suggested URL detector. We achieved better accuracy and F1 record with limited time.

In the study by researchers Ciza Thomas, Sandhya L. and Joby James (2013) several features are compared using different data mining algorithms. The results indicate the efficiency that can be achieved using lexical features. To protect end users from visiting these sites, we can attempt to identify phishing URLs by analyzing lexical and host-based features. The particular challenge in this area is that criminals are constantly developing new strategies to counter our defensive measures. To succeed in this competition, we need algorithms that constantly adapt to new examples and features of phishing URLs. Online learning algorithms provide better learning methods as compared to batch based learning mechanisms [5].

Purbay M. and Kumar D. (2021) [3] proposed a process of classifying phishing attacks according to the scammer's mechanism to trap the alleged users.

Many of these attack methods are master logging tools and DNS disablement. Social engineering startups include online blogs, SMS services, social media platforms that use web services such as Facebook and Instagram, peer-to-peer file-sharing services, and Voice over Internet Protocol (VoIP) systems that attackers use to use caller spoofing identifiers. Each form of phishing differs slightly in how the process is carried out in order to defraud an unsuspecting consumer.

Phishing attacks occur when an attacker sends a message containing a link to potential users to direct them to phishing URLs [4].

3. Methodology

In this study, we use LSTM (Long-Short-Term Memory) which is an algorithm that is part of the structure of our scheme that takes the input from a URL as a character sequence and predicts whether the link is a phishing or a legitimate website [22].

Long and short-term memory is an adaptive and recurrent neural network (RNN), in which each neuron is swapped with a memory cell that is additional to the conservative neuron on behalf of an internal state. Multiplexes are also used as gates to control the flow of information. LSTM layers consist of a set of frequently linked blocks called memory blocks as shown in Fig. 1.

Each of these blocks contains one or more memory cells that are connected repeatedly. Hence, a normal LSTM cell has an input gate that controls the input of data from outside the cell, which determines whether the cell retains or omits data in the internal state, and an output gate that prevents or allows the internal state to be seen from the outside.

The most common is the confusion matrix consisting of four basic scales: true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

Standard metrices, such as accuracy, recall, and F1 score are used in this study to measure the performance of the proposed solutions.

Furthermore, LSTM modules are known to have the ability to learn a large-scale dependency from the input sequence. The LSTM training algorithm uses error gradients to calculate, and combines iterative real-time learning with backpropagation.

However, backpropagation is dropped after the first timestamp because long-term dependencies are handled by memory blocks, not by backpropagation error gradient flow. This step also helped make the LSTM directly comparable to other RNNs in terms of performance because training can be performed using standard backpropagation over time.

LSTM algorithm architecture: The central components of the LSTM architecture are the memory cell, which can maintain its state over time, and the nonlinear gate units, which regulate the information input and output flow of the network as shown in Fig. 2. Based on the insights from secure networks, it is observed that since LSTM neurons consist of internal cells and gate units, one should look not only at the output of the neuron but also at the internal structure to design the original features of the LSTM so that it can address classification problems [25].

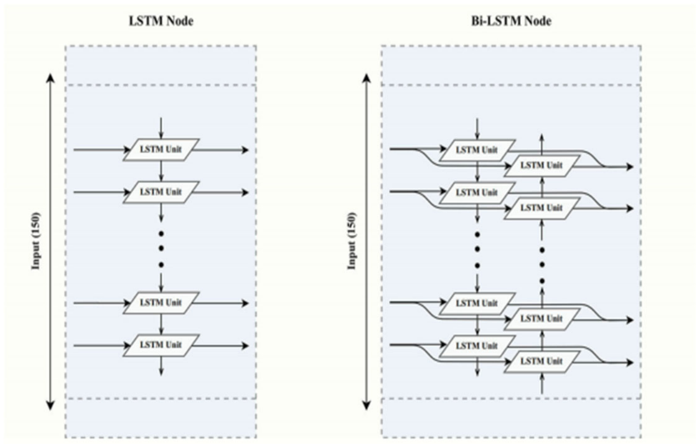

Figure 1. Architecture of the LSTM method – A

Time complexity: In order to calculate the time complexity of the proposed models, the time complexity of the DNN and LSTM based model sections must be calculated separately. For the DNN section, the time complexity is equal to the sum of the number of parameters for each layer because the time is dominated by the matrix multiples of the Multilayer Perception (MLP) layers. As such, the time complexity of the DNN section is O(4p1), where 4 is the number of layers and p1 is the average number of parameters per layer, which depends on the input and output Rate each layer.

The time complexity of each layer in the LSTM is O(1) per weight because the LSTM is local in space and time [26]. Therefore, the time complexity of the LSTM partition is O(w + p2), where the total number of all weights in the LSTM layers and p2 is the number of parameters in the last layer of the LSTM partition. For the model where BiLSTM is used instead of LSTM, the time complexity is O(2w + p2) instead of O(w+p2) because the calculations are done in two different directions in BiLSTM. Due to the structure of the hybrid model, when two separate partitions are combined, two MLP layers are used to obtain the final output value. The time complexity of this final part is O(2p3), where p3 is the average number of parameters in these layers. Finally, the time complexity of the proposed LSTM-based hybrid model is O(w + 4p1 + p2 + 2p3), which is the sum of the time complexity of all parts. Although this combination of models brings an additional cost in terms of time required, the benefit is outside of these additional cost data sets. We did not want to include old datasets and our goal was to conduct experiments in new datasets.

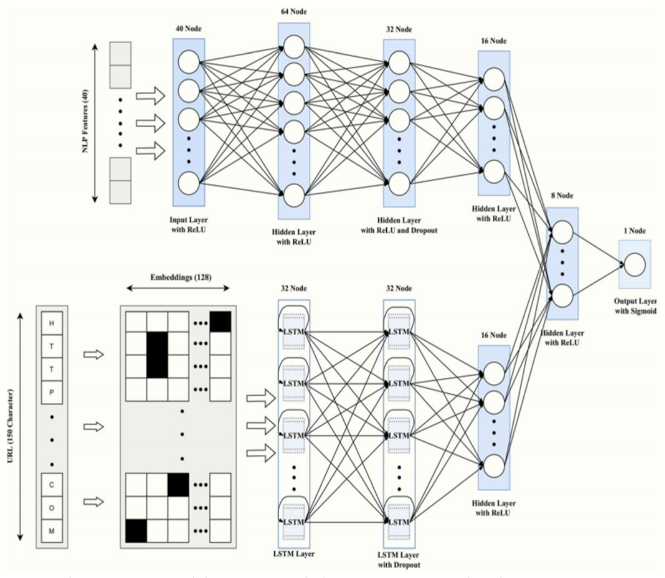

Figure 2. Architecture of the LSTM method - B

3.1 Principles

The proposed framework employs a Recurrent Neural Network (RNN) variant called the Long Short-Term Memory (LSTM) to classify malicious and legitimate website URLs. The RNN implies to broad groups of networks of a similar structure, where on is finite and the other is an infinite input. Both network types contain time dynamic behavior. A recurrent network of finite input is a directed acyclic graph that can be replaced by a purely feedforward neural network, whereas a recurrent network of infinite input is a directed cyclical graph that cannot be modified. The LSTM is a deep learning method, which prevents the gradient problem of RNN. The LSTM comprises multiple gates that are employed to improve the performance. Each input of the LSTM generates an output that becomes an input for the following layer.

a. Recurrent Neural Networks

The recurrent neural network is a type of artificial neural network which uses sequential data or time series data. These algorithms are typically used for temporal or ordinal problems such as speech recognition, image captioning, natural language processing. Similar to the convolutional neural networks and feedforward neural networks, recurrent neural networks utilize training data to learn and optimize the model parameters [30]. The networks' ability to memorize distinguishes them from other machine learning techniques. Such ability is characterized by the way the previous input is used in calculating the weights, which influences the current input and output. Therefore, the recurrent neural network output depends on the sequence of previous inputs. Similarly, the future input can be helpful in determining the output of a specific input, but the unidirectional neural networks cannot account for these events in their prediction [30]. To better understand the concept of recurrent neural networks, the below figure illustrates a simple RNN and how the network can be unfolded to generate the 3rd output as shown in Fig. 3.

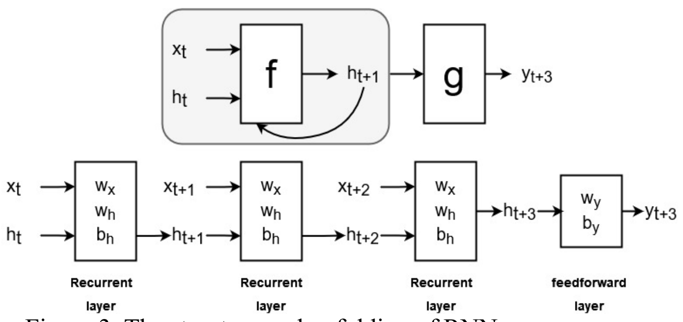

Figure 3. The structure and unfolding of RNN.

As shown in the first part of Fig. 1, the RNN has a feedback loop which is used to unroll in 3 timesteps to produce the second part of the figure. Note that the RNN can be modified to unroll N timesteps as well. While the figure shows a simple illustration of a very small RNN, the topic of RNN is vast and discussing it is beyond the scope of our work. However, we need to discuss the gradient to be able to understand the downsides of the RNN and how the LSTM is used to address these issues.

A gradient is a partial derivative with respect to its inputs, which measures how much the output of a function changes if you change the inputs a little bit. The gradient can be thought of as the slope of a function, where the higher the gradient, the steeper the slope and the faster a model can learn. However, if the slope is zero, the model stops learning. Simply put, the gradient measures the change in all weights with regard to the change in error.

After describing the gradient and what is it used for, we can now discuss the downsides of the RNN. There are two main downsides for the RNN, exploding gradients and vanishing gradients. The exploding gradients happen when the algorithm, unreasonably, assigns high importance to the weights. Fortunately, this problem can be mitigated by truncating or squashing the gradients. Unfortunately, the vanishing gradients problem is not as easy as the exploding gradients. The vanishing gradients problem occurs when the values of the gradients are too small and the model stops learning or takes too long to learn. The problem lasted for a while, but it was solved through the concept of LSTM [31].

b. Long Short-Term Memory

Long Short-Term Memory networks (LSTMs) are an extension for recurrent neural networks, which basically extends the memory. Therefore, it is well suited to learn from important experiences that have very long time lags in between [31]. The units of an LSTM are connected together to be used as the building blocks of the RNN, which is often called the LSTM network. However, building the RNN using LSTM network enables the RNN to remember inputs over a long period of time. The LSTM's ability to remember is due to the LSTM's contains an information memory, which is very similar to the computer's memory. The LSTM can read, write and delete information from its memory. The memory of the LSTM can be described as a gated cell, with the gate controlling whether or not to store or delete the information. The gate opens to store the information or discard the forwarded the information based on the importance of the information, which happens through weights. The weights are learned by the algorithm over time, which simply means the LSTM learns over time what information are important and what are not.

The LSTM have three gates: input gate, forget gate, and output gate. These gates control the memory of the LSTM, where the input gate is responsible of determining whether or not to let new input in, the delete gate is responsible of deleting the information if it is not important, and the output gate allows the information to impact the output at the current timestep. Fig. 4 illustrates a RNN with its gates.

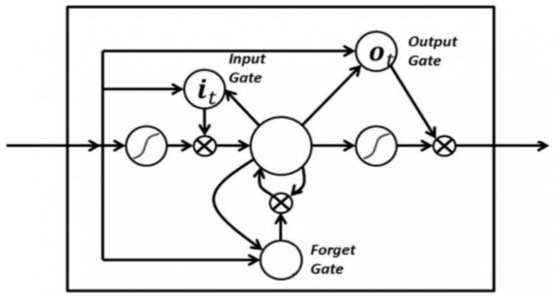

Figure 4. RNN with its three input gates.

The gates in the LSTM are analog and using sigmoid functions, meaning they generate a value that ranges from zero to one. The fact that they are analog enables them to do backpropagation. The problematic issue of vanishing gradients is solved through LSTM as it keeps the gradients steep enough, which keeps the training relatively short and the accuracy high.

After briefly describing the RNN, the problem of vanishing gradients, and how the problem is solved using the LSTM. We now start discussing our proposed design.

3.2 Design

As shown in Fig. 5, we started the process by collecting and preparing the dataset. The dataset was collected from Phish-tank [27]. The used dataset contains 194,798 URLs, of which 97,399 are phishing URLs and the rest is legitimate ones. The data were then split into training and testing datasets. The LSTM network was trained using the training dataset and the performance was evaluated based on the accuracy. The parameters of the LSTM were then modified and tuned to improve the performance before deploying it into the production environment. The classifier can then act as an intermediary stage between the end user and the internet. Whenever a request is sent to any URL, the requested URL is verified using the model and the access is granted if the requested URL is not a phishing URL, or blocked if the requested URL is a phishing URL.

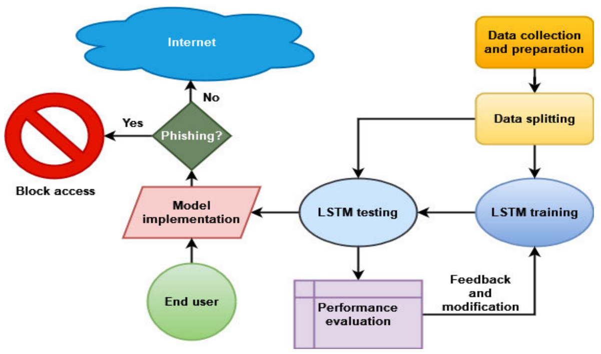

Figure 5. Design of the proposed methodology.

Before diving into the details of the LSTM implementation, we start by discussing the components of the LSTM network layers. The LSTM is an effective prediction and classification mode as it generates an output based on the arbitrary number of implemented steps. The LSTM model contains five essential components that enable the model [28].

Cell State (CS) – a cell that accommodate the long- and short-term memories.

Hidden State (HS) – The output status information that is used to determine the classification based on the current data, input data, and a hidden condition. The HS is used to recover both short-term and long-term memory in order to make the prediction.

Input Gate (IT) – The total number of the information that is fed into the cell state. The input gate identifies an input value for memory alteration. The sigmoid defines the values that ranges from 0 to 1. Then a *tanh* function is used to weight the passed by values to evaluate their significance from -1 to 1. The below equations represent the input gate and the cell state, wherein W_n is the weight, HT_{t-1} is the previous state of the hidden state, xi is the input, and b_n is the bias vector which need to be learnt during the training phase [29].

$$IT = \partial(W_n(HT_{t-1}, x_i) + b_n) \qquad (1)$$

$$CT = tanh(W_d(HT_{t-1}, x_i) + b_c) \qquad (2)$$

Forget Gate (FT) – The total number of data that flows from the current input and past cell state into the present cell state. This gate is used to filter out the information that needs to be discarded from the memory. The sigmoid function is used to describe it contains (HTt-1). The input values (xi) are examined and the number of outputs are verified by each cell state CT_{t-1}.

$$FT = \partial(W_f(HT_{t-1}, x_i) + b_f) \quad (3)$$

Output Gate (OT) – The total number of information that flows into the hidden state. The sigmoid function of this gate determines which values to let through 0 and 1. The tanh function presents weightage of the values which are transferred to determine their degree of importance ranging from -1 to 1 and multiplied with output of sigmoid [29].

$$OT = \partial(W_o(HT_{t-1}, x_i) + b_o) \quad (4)$$

$$HT = O_T * \mathit{TANH}(C_T) \quad (5)$$

3.3 Implementation

The LSTM network is implemented using python with the help of *keras* and *tensorflow* libraries. The LSTM is built using a sequential model and structured as shown in the below figure.

As shown in Fig. 6, the first layer in the LSTM is the input layer, which determines the size and the type of the input into the LSTM network. The second layer is the embedding layer, which sets between the input layer and the LSTM layer. As the LSTM operations are basically floating additions and multiplications, the embedding layer is used to generate a vector float point representation of the input URL. The third layer is the LSTM layer, which utilizes the sigmoid and the tanh functions to adjust the weights, dropout some data which are considered irrelevant, and forward the results into the dropout layer. The fourth layer is the dropout layer, which controls what data flows into the output layer and what data are to be removed from the LSTM memory.

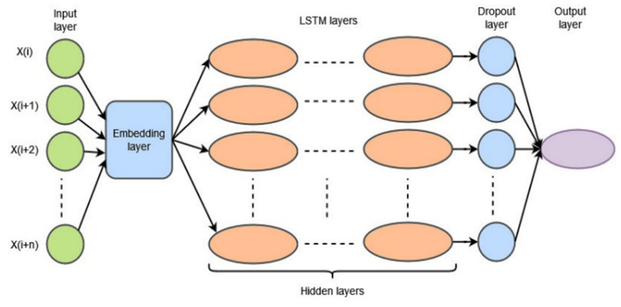

Figure 6. The structure of the LSTM network.

The final layer is the Dense layer or the output layer, which takes the output of the LSTM as an input and produces the classification of the LSTM network. The parameters of the designed LSTM network are shown in the Table 1.

Table 1. LSTM network parameters.

Model	Sequential
Embedding	Input dimension = 100, output dimension = 32, input length = 75
LSTM	Output = 32, dropout = 0.2, recurrent dropout = 0.2
Dense	Activation = sigmoid, Kernel regularizer = regularizers.l2(le-4)
Adam optimizer	Learning rate = 0.0015, loss = binary_crossentropy, metrics = accuracy

4. Discussion and Results

The data set that we used in our research has been well researched and measured by some researchers. The wiki accompanying the dataset comes with a data description document that discusses the data generation strategies taken by the dataset authors [7].

To update our dataset of new phishing sites, we also implemented code that extracts the features of new phishing sites provided by the Phish-Tank website. The dataset contains about 11,000 samples from websites, and we used 10% of the samples in the testing phase. Every website is flagged as legitimate or phishing. The features of our dataset are as follows:

I. Abnormal URL: extracted from WHOIS database. For a legitimate website, the identity is usually part of its URL.

II. Website Redirect Count: If the redirect is more than four times.

III. Web Traffic: This feature measures the popularity of a site by determining the number of visitors.

IV. Page Rank: Page Rank is a value ranging from 0 to 1. PageRank aims to measure the importance of a web page on the Internet.

V. HTTPS Token: Spoof the https token in the URL. For example, http://https-www-mellat-phish.ir

VI. DNS record: DNS record exists.

VII. Request URL: The request URL checks whether external objects in a web page such as images, videos, and sounds have been downloaded from another domain.

VIII. Anchor URL: A link is an element specified by the <a> tag. This feature is treated exactly as the request URL.

IX. Get an IP address: If an IP address is used instead of a domain name in the URL, such as http://217.102.24.235/sample.html.

X. URL length: Scammers can use a long URL to hide the suspicious part in the address bar.

The proposed framework was developed in Python 3.0 using Jupyter notebook software with the support of Numpy, Sci-Kit Learn, Tensorflow, and Keras. To evaluate our proposed framework, we extracted a dataset from the Phish-tank database. The parameters of the training phase of the proposed framework are shown in Table 1. We used a learning rate of 0.0015, which was obtained by sweeping the design space and selecting the best value.

In order to analyze the results of our proposed framework, we used the following metrics:

1) Accuracy: is the percentage of correctly classified URLs.

$$Accuracy = \frac{TP + TN}{TP+FP+FN+TN} \quad (6)$$

2) Recall: is the total number of phishing URLs that are correctly classified.

$$Recall = \frac{TP}{TP+ FN} \quad (7)$$

3) Precision: is the number of correctly predicted phishing URLs.

$$Precision = \frac{TP}{TP+ FP} \quad (8)$$

4) F-measure: is the weighted harmonic mean of the precision and recall of the test. The best value will be at 1 and worst at 0 value.

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad (9)$$

We trained our model using 5 epochs, and we report the results of each epoch as shown in Table 2.

Table 2. The results of the training phase by epoch.

Epoch	Time	Loss	Accuracy
1	195s	0.3562	0.8442
2	192s	0.2709	0.8889
3	194s	0.2415	0.9012
4	195s	0.2224	0.9101
5	191s	0.2104	0.9151

As shown in Table 2, the accuracy of our proposed framework improved by 4.4% from epoch 1 into epoch 2. However, the accuracy improvement decreases to reach 0.5% between epochs 4 and 5. Similarly, the loss decreases by 8.5% between epoch 1 and epoch 2, but drops to a small decrease of 1.2% between epochs 4 and 5. On the other hand, the time to finish running each epoch is slightly affected and hovers around 195 seconds. While the time required by each epoch is expected to be the same, the framework was run on a laptop with the support of Intel® Core™ i7-7700HQ CPU @ 2.8Ghz and 16 GBs of RAM. The Laptop was running multiple applications and have a shared environment as expected, therefore; the time variation can be explained by having a different process interrupting the process, cache misses, processes scheduling… etc.

To test the trained model of our proposed framework, we split the collected dataset into 75% training and 25% for testing. Using the testing part of the dataset, our model reported a loss of 0.1887 and an accuracy of 0.9257. As the results of the testing shows, the performance of our trained model is better than the results reported by the last training epoch. Therefore, we started a sensitivity analysis in which we varied the number of epochs used to train our model and we report the results as shown in Table 3.

Table 3. The performance of our proposed framework with different epochs.

# of epochs	Time of last epoch	Loss of last epoch	Accuracy of last epoch
1	180s	0.3581	0.8440
2	176s	0.2711	0.8886
3	178s	0.2403	0.9026
4	181s	0.2224	0.9100
5	181s	0.2107	0.9153
6	183s	0.2018	0.9193
7	184s	0.1944	0.9221
8	174s	0.1897	0.9246
9	176s	0.187	0.9250
10	176s	0.1829	0.9276

As the results in Table 3 shows, increasing the number of epochs can increase the framework's accuracy and decrease the loss. However, the returns of increasing the number of epochs are diminishing when the number of epochs increases, as we can see from the table, increasing the number of epochs from 9 to 10 increased the accuracy by 0.26% only, and dropped the loss by 0.39% only. Thus. We stopped our sensitivity analysis on the number of epochs at 10 epochs. Testing the trained model at 10 epochs, reported an event better results, which achieved an accuracy of 93.45% and a loss of 16.71%.

To further analyze the results and tune our trained model, we varied the learning rate parameter as shown in the below Table 4.

Table 4. Accuracy with respect to different learning rates.

# of epochs	LR = 0.0015	LR = 0.0001	LR = 0.002
1	0.8440	0.7712	0.8522
2	0.8886	0.8189	0.8964
3	0.9026	0.8289	0.901
4	0.9100	0.8376	0.9160
5	0.9153	0.8439	0.9203
6	0.9193	0.8497	0.9234
7	0.9221	0.8547	0.9258
8	0.9246	0.8589	0.9279

9	0.9250	0.8627	0.9293
10	0.9276	0.8672	0.9307

As we can notice from Table 4, a learning rate of 0.002 achieved the highest accuracy which reached to 93.07%.

The below table shows a comparison between our proposed framework and other frameworks, which used different machine learning techniques to detect phishing websites.

Table 5. Performance comparison between different schemes.

Method	Accuracy	TP rate	FP rate	F-measure
Our framework	93.45%	95.07%	4.9%	93.31%
DNN [32]	88.77%	85.83%	14.17%	-
DNN With GA [32]	91.13%	90.79%	9.21%	-
Decision Table [33]	92.24%	93.2%	6.8%	-
Naïve Bayes [33]	92.98%	93%	7%	-
ANN-MLP [34]	87.61%	-	-	-

As shown in Table 5, our proposed framework has a better performance than other machine learning techniques. We observe that the Naïve Bayes classifier proposed in [33] has the closest accuracy to our proposed framework, which has an accuracy that is 0.5% below the accuracy of our proposed framework. To have a better understanding of our results we show the confusion matrix of our evaluation in Table 6.

Table 6. The confusion matrix of our proposed framework.

		Predictive	
		Positive	**Negative**
Actual	**Positive**	91.60%	4.74%
	Negative	8.39%	95.25%

Table 6 shows the confusion matrix of our proposed framework, we can observe that our scheme has a higher accuracy in correctly detecting legitimate websites. On the other hand, our framework has an accuracy of 91.6% in detecting a phishing URL when it is actually a phishing URL.

Host-based features explain "where" phishing sites are hosted, "who" they are managed by, and "how" they are managed. We use these features because phishing websites may be hosted at less reputable hosting centers, on machines that are not usual web hosts, or through non-reputable registrars.

Using lexical features, we were able to achieve a detection accuracy/success rate of 91.5% for splitting the test at 60%. When using 90% of the data set, we obtained 92.55% detection accuracy. In MATLAB, using a regression tree, we obtained 90.25% detection accuracy when 60% of the data set was used for the test and 87.26% detection accuracy when 90% of the data was used for the test.

5. Conclusion

The proposed framework is an effective technique that addresses the detection of phishing websites by relying on the website's URL. The framework is built using the Long Short-Term Memory algorithm, which improves the Recurrent Neural Networks by solving the diminishing gradients problem. While the problem of phishing cannot be completely removed, however; it can be significantly mitigated by two main ways. First, improving and implementing smart anti-phishing techniques. Second, educating the end users on how fraudulent phishing websites can be detected and identified. To counter the novel and complex phishing attacks and tactics, ML anti-phishing techniques are of extreme importance. In this work, we employed LSTM technique to distinguish malicious and legitimate websites. We used Phish-tank dataset to measure the efficiency of the proposed framework. The results of our evaluation shows that the proposed method presents superior results. A dataset of 194,798 URLs, of which 97,399 are phishing URLs and the rest is legitimate ones. Our framework achieved a very high accuracy in detecting the phishing websites.

6. References

[1] Bahnsen et al. (2018), How to Detect Phishing Website Using Three Model Ensemble Classification.

[2] Preethi, V., Velmayil, G. (2019). Automated phishing website detection using URL features and machine learning technique, International Journal of Engineering and Techniques ,2(5), 107–15. Retrieved 1 Dec 2019, from http://www.ijetjournal.org.

[3] Ashit Kumar Dutta (2021), Detecting phishing websites using machine learning technique, PLOS ONE | https://doi.org/10.1371/journal.pone.0258361 October 11, 2021.

[4] Gunter Ollmann, "The Phishing Guide Understanding & Preventing Phishing Attacks", IBM-Internet Security Systems, 2012.

[5] Sandhya L., Ciza Thomas, Joby James (2013). Detection of phishing URLs using machine learning techniques, International Conference on Control Communication and Computing (ICCC). Publication at: https://www.researchgate.net/publication/269032183. December 2013.

[6] Purbay M., Kumar D (2021), "Split Behavior of Super vised Machine Learning Algorithms for Phishing URL Detection", Lecture Notes in Electrical Engineering, vol.683, https://doi.org/10.1007/978-981-15-6840-4_40.

[7] Mohammad R., Thabtah F. (2016) McCluskey L , (2015) Phishing websites dataset. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites Accessed January.

[8] Adebowale MA, Lwin KT, Sanchez E, and Hossain MA (2019) . Intelligent web-phishing detection and protection scheme using integrated features of Images, frames and text. Expert Systems with Applications, 115: 300-313. https://doi.org/10.1016/j.eswa.2018.07.067

[9] Al-diabat,M. (2016).Detection and Prediction of Phishing Websites using Classification Mining Techniques. International Journal of Computer Applications,147(5) .

[10] APWG (2019). 2nd quarter 2019: Phishing activity trends report.
Anti-Phishing Working Group. https://doi.org/10.1016/S1361-3723(19)30025-9.

[11] Abdelhamid, N., Thabtah, F. and Abdel-Jaber, H. (2017). Phishing detection: A recent intelligent machine learning comparison based on models content and features. In IEEE International Conference on Intelligence and Security Informatics (ISI), 22–77, China:IEEE.

[12] Aburrous, M., Hossain, M., Dahal, K., Thabtah, F. (2010). Experimental case studies for investigating e-banking phishing techniques and attack strategies. cognitive computation, 2(3),242-253.

[13] Dou Z, Khalil I, Khreishah A, Al-Fuqaha A, and Guizani M (2017). Systematization of knowledge (sok): A systematic review of software-based web phishing detection. IEEE Communications Surveys and Tutorials, 19(4): 2797-2819. https://doi.org/10.1109/COMST.2017.2752087

[14] Dunlop M, Groat S, and Shelly D (2014). Goldphish: Using images for content-based phishing analysis. In the 5th International Conference on Internet Monitoring and Protection, IEEE, Barcelona, Spain: 123-128. https://doi.org/10.1109/ICIMP.2010.24.

[15] Nirmal K, Janet B, and Kumar R (2015). Phishing-the threat that still exists. In the International Conference on Computing and Communications Technologies, IEEE, Chennai, India: 139-143. https://doi.org/10.1109/ICCCT2.2015.7292734.

[16] Opara C, Wei B, and Chen Y (2019). HTMLPhish: Enabling accurate phishing web page detection by applying deep learning techniques on HTML analysis. Available online at: https://bit.ly/2zV0ymk.

[17] JainA.K., Gupta B.B. (2018) "PHISH-SAFE:URL Features Based Phishing Detection System Using Machine Learning", CyberSecurity. Advances inIntelligent Systems and Computing, vol.729,https://doi.org/10.1007/978-981-10-8536-9_44.

[18] Luke ,I. (2020).The 5 most common types of phishing attack.[online] Retrieved 1 March 2020, from https://www.itgovernance.eu/blog/en/the-5-most-common-types-of-phishing-attack.

[19] Adebowale MA, Lwin KT, Hossain MA (2020) Intelligent phishing detection scheme using deep learning algorithms. J En-terprise Inf Manag.

[20] Akinyelu AA (2019) Machine learning and nature inspired based phishing detection: a literature survey. Int J Artif Intell Tools 28(05):1930002.

[21] Wei, W.; Ke, Q.; Nowak, J.; Korytkowski, M.; Scherer, R.; Wo´zniak, M. (2020) Accurate and fast URL phishing detector: A convolutional neural network approach. Comput. Netw, 178, 107275. [CrossRef]

[22] Chen, D.; Wawrzynski, P.; Lv, Z. (2021) Cyber security in smart cities: A review of deep learning-based applications and case studies.Sustain. Cities Soc. 66, 102655. [CrossRef]

[23] Al-Ahmadi, S. PDMLP: Phishing Detection Using Multilayer Perceptron. Int. J. Netw. Secur. Its Appl. 2020, 12. SSRN:3624621. Available online (accessed on 12 May 2021) : https://papers.ssrn.com/abstract=3624621.

[24] Ahmad, R.; Alsmadi, I. (2021) Machine learning approaches to IoT security: A systematic literature review. Internet Things , 14, 100365.

[25] Dargan S, Kumar M, Ayyagari MR, Kumar G (2020) A survey of deep learning and its applications: a new paradigm to machine learning. Arch Comput Methods Eng 27(4):1071–1092

[26] Hao S, Ge FX, Li Y, Jiang J (2020) Multisensor bearing fault diagnosis based on one-dimensional convolutional long short-term memory networks. Measurement 159:107802

[27] Phishtank website," https://phishtank.org/", accessed: 2022-02-06

[28] Dutta, Ashit Kumar. (2021) "Detecting phishing websites using machine learning technique." PloS one 16, no. 10: e0258361.

[29] LSTM structure and gates," https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn", accessed: 2022-02-06.

[30] Recurrent Neural Networks," https://www.ibm.com/cloud/learn/recurrent-neural-networks", accessed: 2022-02-06.

[31] A guide to RNN understanding," https://builtin.com/data-science/recurrent-neural-networks-and-lstm", accessed: 2022-02-06.

[32] 32. Ali, Waleed, and Adel A. Ahmed. (2019) "Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting." IET Information Security 13, no. 6: 659-669.

[33] Abdulrahman, Musbau Dogo, John K. Alhassan, Olawale Surajudeen Adebayo, Joseph Adebayo Ojeniyi, and Morufu Olalere. (2019) "Phishing attack detection based on random forest with wrapper feature selection method.".

[34] Ferreira, Ricardo Pinto, Andréa Martiniano, Domingos Napolitano, Marcio Romero, Dacyr Dante De Oliveira Gatto, Edquel Bueno Prado Farias, and Renato José Sassi. (2018) "Artificial neural network for websites classification with phishing characteristics."