Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

# Detecting phishing attacks using a combined model of LSTM and CNN

Subhash Ariyadasa [1, 2, *], Subha Fernando [1], Shantha Fernando [3]

[1]*Department of Computational Mathematics, University of Moratuwa, Moratuwa, Sri Lanka*
[2]*Department of Computer Science and Informatics, Uva Wellassa University, Badulla, Sri Lanka*
[3]*Department of Computer Science and Engineering, University of Moratuwa, Moratuwa, Sri Lanka*

ARTICLE INFO

*Article history:*
Received 10 December 2019
Received in revised form
30 March 2020
Accepted 1 April 2020

*Keywords:*
Phishing
LSTM
CNN
Cybersecurity

ABSTRACT

Phishing, a social engineering crime which has been existing for more than two decades, has gained significant research attention to find better solutions to face against the very dynamic strategies of phishing. The financial sector is the primary target of phishing, and there are many different approaches to combat phishing attacks. Software-based detection approaches are more prominent in phishing detection; however, still, there is no robust solution that can stable for a long period. The primary purpose of this paper is to propose a novel solution to detect phishing attacks using a combined model of LSTM and CNN deep networks with the use of both URLs and HTML pages. The URLs are learned using an LSTM network with 1D convolutional, and another 1D convolutional network is used to learn the HTML features. These two networks were trained separately and combined through a sigmoid layer by dropping the last layer of each model to have the proposed model. The proposed model reached 98.34% in terms of accuracy, and that is above the previously recorded highest accuracy of 97.3% among the detection models used both URL and HTML features in the explored literature. The solution requires feature extraction only with HTML pages, and URLs were directly fed with a minimum pre-processing. Although the proposed solution uses extracted HTML features, those do not depend on third-party services. Therefore, an efficient real-time application can be implemented using the proposed model to detect phishing attacks to safeguard Internet users.

## 1. Introduction

Phishing, which is originated from the term fishing, is defined as impersonating a trusted third party to steal personal and confidential information from a victim (Whittaker et al., 2010). It was started in 1995 with the American Online (AOL) attack (Chiew et al., 2018b) and still exists as a significant cyber threat by having a top rank in the cyber threat landscape (ENISA, 2019). Phishing is highly associated with human intellect (Nirmal et al., 2015), and the financial gains are the primary motivation for this kind of attack. However, fame and notoriety is also an exciting psychological aspect of phishing (Weider et al., 2008). Phishing is a severe security problem today, and phishers are smart, economically motivated, and adaptable. The European Union Agency for Cybersecurity (ENISA) is ranked phishing within the top 4 out of 15 top cyber threats (ENISA, 2019).

Further, the Anti-Phishing Working Group (APWG) also identified more than 180,000 unique new phishing sites for the second quarter of 2019 (APWG, 2019). According to the APWG, nearly 22% of phishing attacks found in online payment systems, and next is the financial sector, and it is 18%. That means more than 40% of phishing attacks were reported in payment processors and banks. However, as a new trend, nearly 39% of attacks also reported in Software as a Service (SaaS) and cloud storage. All these facts claim that phishing is still an active threat.

In literature, there are different approaches to combat phishing attacks and, those are mainly categorized under two, namely, improving user-awareness and software-based detection. However, the second approach, software-based detection, which is also used in this study, is having a high potential interest because it is a human-centric approach. There are different software-based detection approaches; among those, machine

* Corresponding Author.
Email Address: 188077d@uom.lk (S. Ariyadasa)
https://doi.org/10.21833/ijaas.2020.07.007
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-7937-128X

learning performs well due to the unique advantages of it.

Deep learning, a representation learning approach, is dominated the Artificial Intelligence (AI) field for the past few years (LeCun et al., 2015). It is very good at discovering complex structures in high dimensional data; therefore, deep learning applies to many domains, and it only requires minimal engineering by hand (LeCun et al., 2015). The study is also based on two well-known deep learning techniques; Long/Short Term Memory (LSTM) and Convolutional Neural Network (CNN). The proposed model used these two techniques to detect phishing attacks using both HTML and URL based features. The LSTM and 1D convolutional network are used to learn abstract level features in URLs by getting the website URL as the input. The specialty here is the URLs are used without any manual feature extraction. Those are directly fed to the network with a minimum pre-processing. The HTML features which extract from HTML pages through a feature extraction model are separately trained in a 1D convolutional network. Finally, the knowledge of both these two networks is combined through a dense layer with a sigmoid activation function to make final predictions. The proposed solution shows an average accuracy of 98.3% in detecting phishing attacks, and it is the highest recorded accuracy in a model which implemented using both URL and HTML features in the explored literature. The main contribution of this paper is a new deep network to detect phishing attacks in higher accuracy using both HTML features and URLs.

The rest of the paper is organized as follows. In Section 2, the paper discusses the overview of phishing and the detection approaches used in the past. Section 3 describes the proposed solution, and Section 4 explains how the experiment was done. Then the results obtained and the performance of the proposed solution is presented in Section 5. Finally, in Section 6, the paper concludes by mentioning some future directions.

## 2. State of the art

Phishing, the Internet-based attack or cyber-attack which exists for more than two decades now (Chiew et al., 2018b), is an attempt by an individual or a group of people to steal personal or confidential information of a victim (Nguyen et al., 2014a). It is a social engineering crime (Whittaker et al., 2010), which is having a growing tendency during the last two decades (Li et al., 2019; APWG, 2019). Li et al. (2019) mentioned 1609 phishing attacks per month, which is now increased to more than 50,000 attacks per month in the 2nd quarter of 2019 (APWG, 2019). The main reason behind such a tendency is the nature of the phishing attacks because these attacks do not remain for more extended periods; suddenly come and get the work done, then disappears. However, the complexity, confusing and, noising of these attacks make it hard to detect and challenge the researches to find a robust solution.

### 2.1. Overview of phishing attacks

Phishing attacks mainly used three strategies, namely, mimicking attack, forward attack, and pop-up attack (Chiew et al., 2018b). Mimicking attacks are frequent and used emails to send a fake URL to the victim as bait (Chiew et al., 2018b). Generally, phishing attacks are started with an impersonated legitimate web page (Li et al., 2019), which is very much similar to the legitimate web page (Adebowale et al., 2019). Further, phishing attacks consist of three main components as the medium of phishing, attack vector, and technical approaches (Chiew et al., 2018b). The medium of phishing can be the Internet, which is more popular, SMS, or Voice. Attack vectors are Email, Instant Message (IM), Social Networking, Website, and more. The technical approaches which are used to enhance the attack further are mainly two types; vulnerability exploitation on hardware or software and website related techniques, which is more prevalent in phishing (Chiew et al., 2018b).

Generally, a phishing attack is executed in six main steps: 1) the attacker constructs a fake website by finding a target brand and audience, 2) the URL of the fake website distributes to the audience through numerous spam emails, 3) user reads the email and act (i.e., click on the link) on it, 4) the user interacts with the fake website, 5) the attacker collects sensitive information, and 6) the collected information is used to satisfy attacker's intention. However, the life of a phishing cycle is concise, and half of the phishing attacks are being shut down in less than a day. Further, the average uptime of a phishing web page is 32.5 hours, as stated in the literature (Li et al., 2019).

### 2.2. Phishing detection approaches

Many methods have been developed to safeguard users from phishing attacks. Email filtering and web page or deceptive phishing detection are standard methods for such attack detection (Dou et al., 2017). However, the current study is primarily focused on web page phishing detection; therefore, the email-based phishing filtering is not included in this paper as a phishing detection approach. In the past two decades, different technical and non-technical anti-phishing solutions introduced to the community, and those solutions mainly into two categories; improving user-awareness and software-based detection (Khonji et al., 2013).

#### 2.2.1. Improving user-awareness

Phishers are always taking advantage of inexperienced users to accomplish their intentions, and improving user-awareness is one solution to overcome this (Khonji et al., 2013). Dong et al. (2008) proposed a visual user phishing interaction model, which helps to identify the failures of users when interacting with the websites. The Anti-Phishing Phil (Sheng et al., 2007) was another solution introduced to practice good user habits in an interactive gaming environment. Similarly, Smells Phishy (Baslyman and Chiasson, 2016) was a game-based attempt to improve user-awareness. Displaying warnings and notifications to the users are common in many browsers today, and the use of active warning rather than passive gives superior results in improving user-awareness (Egelman et al., 2008; Wu et al., 2006). Further, the training materials can be used to improve user-awareness (Khonji et al., 2013). Although improving user-awareness shows some success, it is a machine-centric approach which is not practical and effective in the phishing domain (Khonji et al., 2013).

### 2.2.2. Software-based detection

A software-based detection is a human-centric approach that can be categorized into four categories, namely, blacklisting/whitelisting, rule-based heuristic, visual similarity, and machine learning (Khonji et al., 2013).

Blacklisting/whitelisting Techniques: A simple and commonly used approach depends on a list of phishing or legitimate web site URLs. Known phishing URLs list is referred to as blacklist and whitelist stores legitimate ones (El-Alfy, 2017). Google Safe Browsing API (https://safebrowsing.google.com/) is one such blacklist used in the present. Even though this is a simple approach, maintaining a black or white list mainly depends on reporting and confirmation of suspicious websites, which requires more time and effort (Jain and Gupta, 2016). Further, practical limitations such as the need for exact matching, failures in detecting zero-hour attacks, and maintaining an up-to-date list (Khonji et al., 2013; Jain and Gupta, 2016; El-Alfy, 2017), make this approach ineffective. PhishNet tool (Prakash et al., 2010), Automated Individual White-List (AIWL) (Cao et al., 2008), and White-List maintainer (Jain and Gupta, 2016) are few approaches used to overcome from some of the mentioned issues.

Rule-based Heuristic Techniques: This technique can detect zero-hour attacks (Khonji et al., 2013). However, as Khonji et al. (2013) stated, the risk of misclassifying legitimate websites is also high in this technique. SpoofGuard (Chou et al., 2004), uses a set of rules based on the features like domain name, URL, links, and images to detect phishing attacks. CANTINA (Zhang et al., 2007), a content-based approach, used the TF-IDF algorithm with six heuristics like age of the domain, known images, IP Address, and few more. CANTINA performs well compared to the SpoofGuard by having 90% accuracy, with only 1% of the false-positive rate (Zhang et al., 2007). PhishGuard (Joshi et al., 2008), another heuristic approach that is based on the HTTP digest authentication concept, used HTTP 200 OK and 401 unauthorized statuses when detecting phishing attacks. Similarly, Mohammad et al. (2014a) proposed an intelligent rule-based technique with 17 selected features. Although the rule-based heuristic approaches have good detection accuracy, problems such as high False Positive (FP) rate, predefined rules, cost of updating rules, and rapidly changing nature of phishing attacks (Khonji et al., 2013; Gupta et al., 2017) make this also ineffective.

Visual Similarity Techniques: Visual similarity techniques have used the appearance of the web page and mostly features like text content, text format, HTML tags, CSS, images, and more. DOMAntiPhish (Rosiello et al., 2007), one such technique, uses the Document Object Model (DOM) similarity between two pages through a defined function in detection. Nguyen et al. (2014b) proposed another DOM tree-based approach to overcome the arouse issues in Rosiello's approach through a two-way similarity comparison technique. PhishZoo (Afroz and Greenstadt, 2011), which used a profile based technique with an accuracy of 96.1%, is used URL of the website, SSL certificate, and web content like HTML, images, and scripts. It is a profile based technique.

Similarly, Huang et al. (2010) proposed a site signature approach, which creates a unique web-based signature using text and image-based features, and it shows 94% accuracy with a low error rate. Goldphish (Dunlop et al., 2010) is having the ability to detect zero-hour phishing attacks and shows better results compare to previous solutions. However, this solution is unstable because it depends on the logo image, OCR, and Google ranking (Adebowale et al., 2019; Jain and Gupta, 2017). Phishing-Alarm, a Cascading Style Sheet (CSS) based solution (Mao et al., 2017), uses CSS as the basis to measure the visual similarity. Likewise, several other approaches in visual similarity area like discriminative key point features which have a high degree of accuracy between 95% and 97% (Chen et al., 2009), Earth Mover's Distance (EMD) which works at the pixel level of the web pages with significant precision (Fu et al., 2006) and hybrid approaches in phishing detection (Jain and Gupta, 2017) are also mentioned in literature. However, problems like accuracy issues, use of databases, failures in zero-hour attacks, embedded objects detection issues, and use of threshold value (Jain and Gupta, 2017) are mentioned as drawbacks of this technique.

Machine Learning Techniques: An association rule mining approach was proposed in phishing detection by Jeeva and Rajsingh (2016). They had been used fourteen heuristic rules to extract features from URLs, and a total of 18 rules were generated to achieve 93% accuracy. Nguyen et al. (2014a) used six heuristics with a single-layer neural network to achieve 98% of accuracy. Although Nguyen et al. (2014a) achieved good accuracy, some of the used heuristics highly depend on third-party services. Phish-Safe (Jain and Gupta, 2018), which is based on Support Vector Machine (SVM), used 14 features and achieved the best detection accuracy of 90%. Sahingoz et al. (2019) compared seven different machine learning algorithms with three different feature vectors like word, Natural Language

Processing (NLP) based, and hybrid to detect phishing URLs. The result shows that the Random Forest (RF) algorithm with NLP based features gives the best accuracy of 97.98%. Further, Probabilistic Neural Networks (PNNs) is used by El-Alfy (2017) to implement a classifier with 96.74% of detection accuracy. Although these mentioned approaches show some accuracy above 90%, all these approaches only depend on URLs and suffering from manual feature extraction.

As a solution for this manual feature extraction, deep learning techniques were tried out to implement automated feature extraction processes in the past. HTMLPhish (Opara et al., 2019) was such an attempt that used Recurrent Neural Network (RNN) to automated feature extraction process from HTML pages. It used only HTML pages in the detection process and achieved 97.2% detection accuracy. Further, Bahnsen et al. (2017) proposed an LSTM network-based solution with high precision. The solution only used URLs, and no manual feature extraction is required. The URLs were fed to the LSTM network after an encoding process, and it reduces the detection time. That was the first time LSTM was used in phishing detection, and it outperformed with 98.7% accuracy. After that, Chen et al. (2018) also used LSTM to detect phishing URLs, and they have achieved 99.1% of accuracy.

Further, Chen et al. (2018) reported that the CNN approach with the URLs has less accuracy compared to the LSTM. However, Pham et al. (2018) stated that a combination of CNN and LSTM could give better results in detecting malicious URLs rather than using only LSTM. Although high accuracy is maintained in these automated malicious URL detection systems, URL shortening services that can hide malicious URLs, benign URLs becoming malicious in the future, and tools which can simulate URLs to bypass these models can be a challenge to have an effective phishing detection in the long run (Sahoo et al., 2017).

To overcome such challenges, incorporating HTML features extracted from the web page content with URL features in phishing detection is a strategic approach which also studied in the literature. A self-structuring multilayer perceptron network was proposed to detect phishing attacks by Mohammad et al. (2014b) with 17 input features, including both HTML and URL features. The solution achieved 92.5% of detection accuracy. Similarly, Pratiwi et al. (2018) also proposed a neural network architecture with 18 input features with a low accuracy rate of 83.38%. Li et al. (2019) used Gradient Boosting Decision Tree (GBDT), XGBoost, and LightGBM in multiple layers with 8 URL and 12 HTML based features. That is the first stack model to detect phishing attacks and achieved 97.3% accuracy. Further, Subasi et al. (2017) used several machine learning algorithms in phishing detection, and out of all, RF outperformed with an accuracy of 97.36%. However, no one in the explored literature tried to incorporate HTML features with LSTM approached introduced by Bahnsen et al. (2017) to experiment whether it can provide a robust solution to overcome this social engineering crime.

## 3. Proposed solution

The overview of the proposed solution to detect phishing attacks is shown in Fig. 1. The data source contained URLs and HTML codes of web pages. The URLs are directly used as inputs to the model with a minimum pre-processing, and that is separately discussed in a below subsection. However, HTML features need to be extracted from the web pages. Therefore, a feature extraction model is used for the extraction before finalizing the model input features. After extracting the relevant features from the web pages, HTML features, and URLs concatenate to have input feature vectors for the detection model. Finally, the detection model will use the input feature vector and produce an output as legitimate or phishing. However, the detection model is a combination of two deep networks. It can analyze URLs and HTML features separately and combine both decisions in making the final output of the model. The major components included in the solution, namely, a feature extraction model and detection model, are introduced in the following subsections.

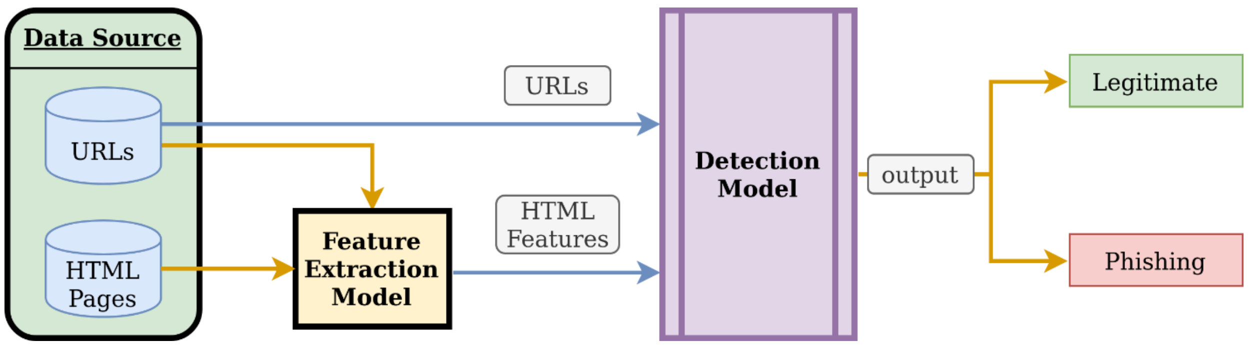


**Fig. 1:** The proposed solution to detect phishing attacks

### 3.1. Feature extraction model

The URLs are directly used as inputs to the detection model after performing a minimum pre-processing on it. Therefore, the feature extraction model used only to extract HTML features. However, Fig. 1 shows that the URL is also used as an input to the feature extraction model. That is only to extract the website domain name to support the HTML feature extraction process. The model will extract 15 HTML features from a given web page, and those features are described below:

- **Number of hyperlinks** (Jain and Gupta, 2016): Number of 'href' attributes relevant to <a> in a web page.
- **Number of null pointers** (Jain and Gupta, 2016; Gu et al., 2013): Number of 'href' attributes with the value empty or '#' on a web page.
- **External link ratio** (Gu et al., 2013; Jain and Gupta, 2016; Li et al., 2019): Ratio between total number of available hyperlinks and external links.
- **Personal data forms** (Li et al., 2019; Gupta et al., 2017): Binary value is used to check whether a <form> tag with one or more <input> child tag available in a page.
- **Length of the HTML page** (Li et al., 2019): HTML code will be taken as a string and calculate the length of it.
- **Number of <script> tags** (Li et al., 2019): The number of <script> tags used in the web page.
- **Number of <link> tags** (Li et al., 2019): The number of <link> tags used in the web page.
- **Number of <!–> tags** (Li et al., 2019): The number of comments used in the web page.
- **External resource ratio** (Li et al., 2019): Ratio calculate using HTML tags like <img>, <script>, and <link>.
- **Favicon** (Chiew et al., 2019): Binary value is used to indicate whether a web page is having a favicon and loaded from the same domain.
- **Internal form ratio** (Chiew et al., 2019): Ratio between the available <form> tags and the number of form's action attribute has the same domain or relative path.
- **Abnormal form ratio** (Chiew et al., 2019): Ratio between the available <form> tags and the number of form's action attribute contains a '#', 'about: blank' or an empty string.
- **External form ratio** (Chiew et al., 2019): Ratio between the available <form> tags and number of form's action attribute contains a URL from an external domain.
- **Title tag** (Chiew et al., 2019): Binary value is used to check whether <title> tag is used one time on the page inside the head area.
- **Title tag and brand name** (Li et al., 2019): Binary value is used to check whether the <title> tag contains the URL brand name.

### 3.2. Detection model

The detection model consists of three sub-models, as shown in Fig. 2. The two sets of features mentioned above, URL and HTML features, are used in the detection model. These two sets will train separately with two deep learning models and merged the outputs of the models with the concept of transfer learning to build the final model. Then the final model will train again with both sets of features and used directly to identify the phishing and legitimate web pages. The procedure of the proposed detection model is summarized in Table 1, and three sub-models will introduce intensely in the following subsections.

**Table 1:** Steps of the proposed model to detect phishing attacks

**Step 1:** Construction of the Data for the Model
- URL will take as one input feature
- HTML features will be extracted after going through a feature extraction model
- Combine the URL and HTML features to construct the final input feature vector
- Output label associate to the input feature vector will merge and create an input to the model

**Step 2:** Division the Model Input into Input Vectors
- Input vector one is created with URL and associated output label
- Input vector two is created with HTML features and the output label

**Step 3:** Model A Training
- Input vector one is used with the 1D convolutional and LSTM model
- URLs are pre-processed and used to train the model
- Model is trained and saved on the disk

**Step 4:** Model B Training
- Input vector two is used with the 1D convolutional model
- Model is trained and saved on the disk

**Step 5:** Model C Training
- Model A is loaded from the disk and remove the last sigmoid layer
- Model B is loaded from the disk and remove the last sigmoid layer
- Last output layers of Model A and B concatenated and used as the input for the Model C
- Model C is trained and use a test set to evaluate the model

**Step 6:** Make Predictions from the Model
- Model input will be created with the unseen web page by following the first three procedures of step 1
- The input will pass to the Model C
- Model C will output whether the web page is phishing or legitimate

#### 3.2.1. Model A: 1D convolutional and LSTM model

LSTM is proven to be that it is a powerful technique for detecting phishing URLs (Bahnsen et al., 2017; Chen et al., 2018). Further, Pham et al. (2018) have shown that the combination of 1D convolution layer and LSTM layer improves the accuracy, compared to the models that consider only LSTM layers in malicious URL detection. Therefore, this study selected 1D convolutional and LSTM architecture to train the URL features when designing the Model A.

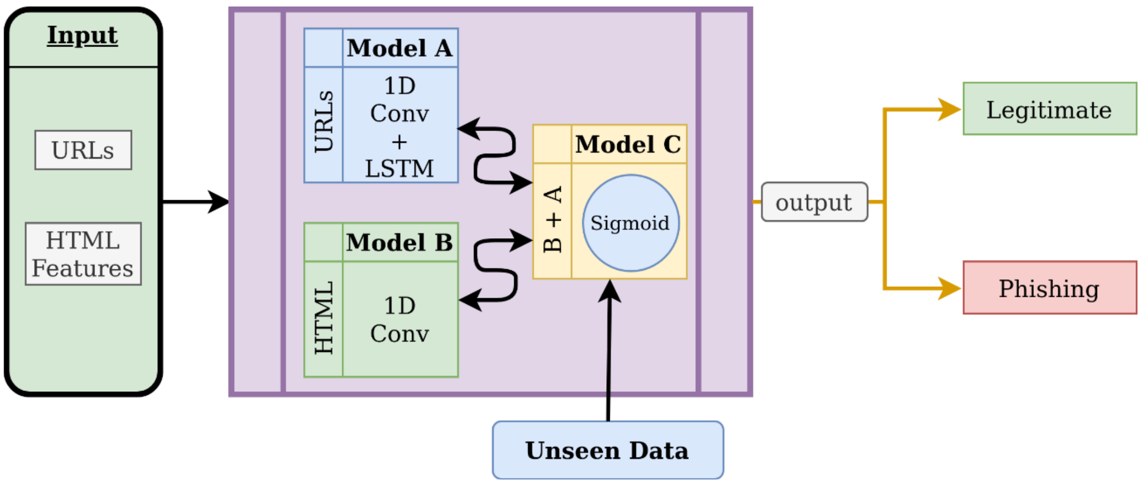


**Fig. 2:** Overview of the detection model

In this work, first, pre-processing of the URL is required. Each character of the URL was considered as a word and gave a unique integer value to those words using Python's printable class in the string package. It is sufficient at this level since all the selected URLs are in English. Then to make all URLs in the same size, URLs were chopped into one size, and the size was decided by analyzing the URLs' character length distribution. Fig. 3 shows the URL character length distribution for legitimate and phishing URLs. Therefore, the maximum URL character length was selected as 150, and the URLs which had lesser characters were padded with 0.

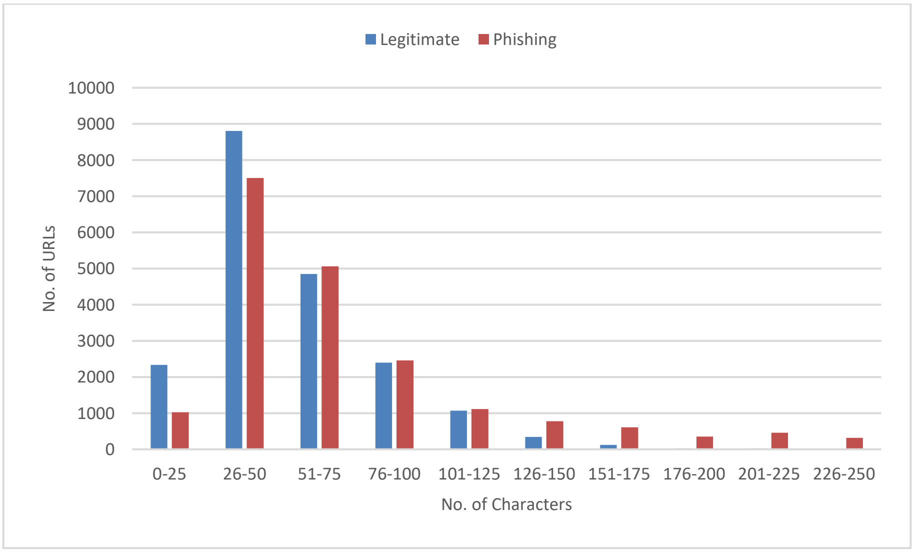


**Fig. 3:** Character length distribution of the URLs

Model A was designed as a feed-forward network, and it contains an input layer, embedding layer, 1D convolution layer, pooling layer, LSTM layer, and output layer. Pre-processed URLs are passed as inputs to the model, and it defines the initial input shape. Then the input character is translated by a 256-dimension embedding in the embedded layer. Next, the translated URLs are fed into the 1D convolution layer through a chaining approach, and the layer uses ReLU as the activation function. Then as a common approach, the pooling layer is used at the end of the convolution part. The output of the convolution part is fed next to the LSTM layer, which is having a hyperbolic tangent (tanh) activation function with an output size of 32. The output layer of the model is designed with a dense layer with one neuron and sigmoid activation function, and it is where the actual classification takes place; therefore, the LSTM layer output is fed to the output layer to perform the classification task. The network uses binary cross-entropy as the loss function with Adam optimizer, and dropouts are used in each hidden layer. Fig. 4 shows a summary of model A.

### 3.2.2. Model B: 1D convolutional model

Model B is designed to train the HTML features, and it is a simple 1D convolutional network. It also uses a multilayer perceptron approach and contained an input layer, two 1D convolution layers, pooling layer, flatten layer, dense layer, and output layer. The inputs are first converted to a floating-point value and pass to the model for the shaping. Then input goes through two 1D convolution layers, which used ReLU as the activation function. Then the pooling and flatten layers are activated and passed the output to a dense layer, which has 32 neurons. The dense layer uses ReLU as the activation function, and the output of the layer is fed to the output layer of the model, which is also a dense layer with one neuron and sigmoid activation function. Similar to the Model A, Model B also uses binary cross-entropy as the loss function with Adam optimizer, and dropouts are used after each convolution layer. Fig. 5 shows the summary of the model B.

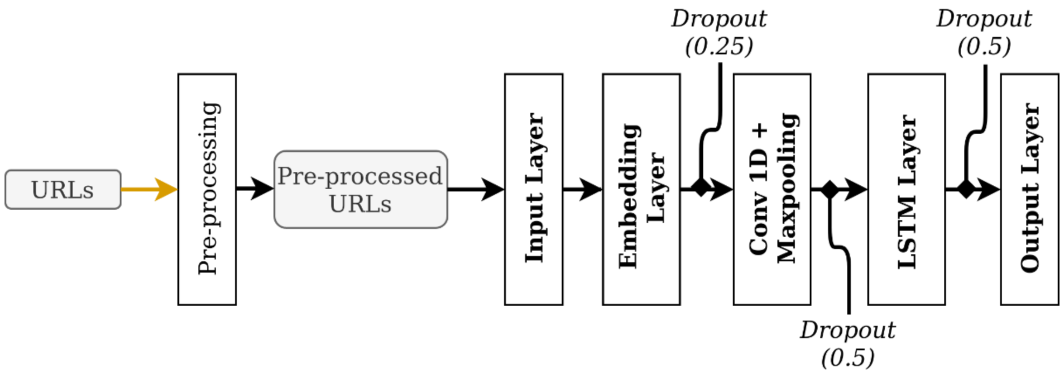


**Fig. 4:** Proposed network architecture for model A

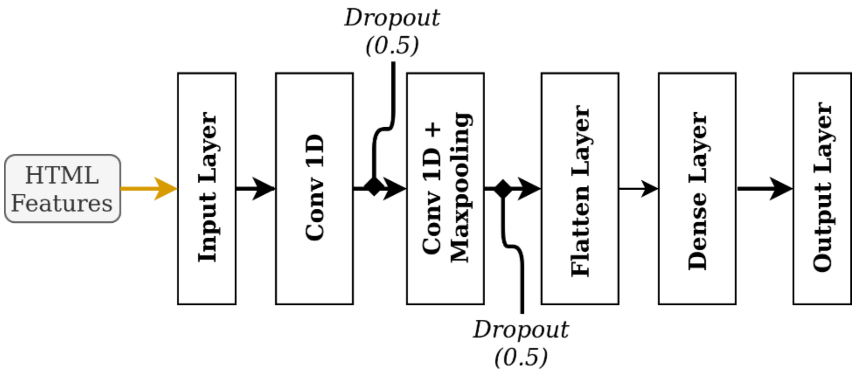


**Fig. 5:** Proposed network architecture for model B

### 3.2.3. Model C: Prediction model

Model C is designed with the concept of transfer learning. Model A and B are separately trained and load to the Model C. Then, the output layers of Model A and B are removed. Then the final layer of the Model A is the LSTM layer, and Model B is the dense layer. Both final layers have 32 outputs each, and those outputs are concatenated to use as input to the Model C. Model C is a simple network with one dense layer. The layer has one neuron, and it uses the sigmoid activation function. After sufficient training, Model C is used for the prediction task.

## 4. Experiment and evaluation

The experiment is performed on an HP ProBook machine with 8 GB of memory, an Intel Core i5-7200U CPU @ 2.50GHz x2 processor. Keras neural-network library on top of TensorFlow and Python programming language, are used in all implementation tasks.

### 4.1. Data source

The experiment used a self-constructed data source with 40000 data. The data source consisted of 20000 legitimate and 20000 phishing web pages with relevant URLs. The legitimate web pages were collected from the Google search engine through a Python script. The script can handle the duplicates, and the top-ranked web pages were selected based on the Google page ranking to have a trusted, legitimate set. Further, the script used a word list from GitHub and a self-generated list while executing the searching task. The phishing web pages with URLs were collected from several sources, mainly, PhishTank (https://www.phishtank.com/) and the phishing web site data source (Chiew et al., 2018a) of the University Malaysia Sarawak available in the University official link (http://www.fcsit.unimas.my/research/legit-phish-set/).

Further, the data collected except PhishTank were verified using either PhishTank or Google Safe Browsing API to construct an accurate phishing data source. Therefore, all the data used in the phishing data source are either available in PhishTank or Google Safe Browsing API or both. The final data source was constructed in CSV format after the feature extraction model was extracted 15 HTML features, by merging relevant URLs and class labels. Then the CSV file, which contains 17 columns (15 HTML features + URL + class label) and 40000 rows, were divided randomly using the scikit-learn python library to have three separate data sources for training, testing, and validation. The proportions used for training, testing, and validation are 70%, 20% and, 10%, respectively.

## 4.2. Performance metrics

Phishing detection is a classification problem. Therefore, the confusion matrix approach is the best way to summarize the predictions to evaluate the performance of the proposed solution. The confusion matrix relevant to the study is shown in Fig. 6.

| **Predicted** \ **Actual** | Phishing (1) | Legitimate (0) |
|---|---|---|
| Phishing (1) | TP | FP |
| Legitimate (0) | FN | TN |

**Fig. 6:** Confusion matrix used during the study

Each feature vector is fallen into one of the four possible categories mentioned in Fig. 6. The True Positive (TP) category contains the correctly predicted phishing pages, and True Negative (TN) is for correctly predicted legitimate pages. Then, False Negative (FN) and False Positive (FP) are the categories where the incorrect classification is happening. The FP contains legitimate pages predicted as phishing, and in FN, phishing pages are predicted as legitimate. Phishing detection is highly sensitive to false positives because if a single prediction falls into that category may cost more due to the nature of the phishing attacks.

The standard measures, such as accuracy, precision, recall, and f1-score, are used in this study to evaluate the proposed solution's performance. The mentioned metrics are described in the Eqs. 1-4.

$$Accuracy = \frac{TN+TP}{TN+FP+TP+FN} \quad (1)$$

$$Precision = \frac{TP}{FP+TP} \quad (2)$$

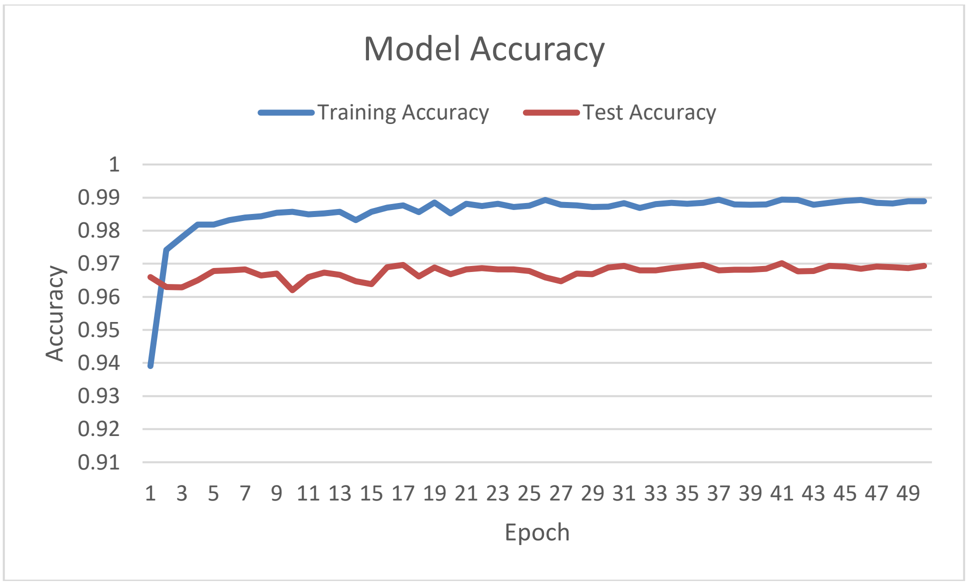
$$Recall = \frac{TP}{FN+TP} \quad (3)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad (4)$$

Further, the Receiver Operating Characteristic (ROC) curve, which is useful when predicting the probability of a binary classification task, is also used with Area Under the Curve (AUC) to evaluate the proposed solution's performance statistically.

## 4.3. Training and evaluation

Model A and B were trained separately for 100 epochs with a batch size of 64 under 0.001 learning rate and saved to the disk. Then the training of Model C was started. It trained in a 50-step sequence with a learning rate of 0.001. The three data sources mentioned above were used in the experiment, and the training source was used for training, and the test source was used for internal validation. Fig. 7 shows the final model accuracy and loss, respectively, in each epoch for both training and testing data sources. After analyzing the graphs, it was shown that the performance on a validation data set starts to degrade before ten epochs. That is an indication of an overfitting scenario. Therefore, the early stopping technique was used to stop the training of the model early before it has overfitted the training data set. After the model successfully fit, 10% of data reserved for validation was used to evaluate the model performance. Model C was trained and evaluated three times using different data set for each time in the same proportions as mentioned above for training, testing, and validation to have a less biased model at the end. The scikit-learn model selection is used with different random states in this task. The results obtained through the experiments are discussed in the next section.



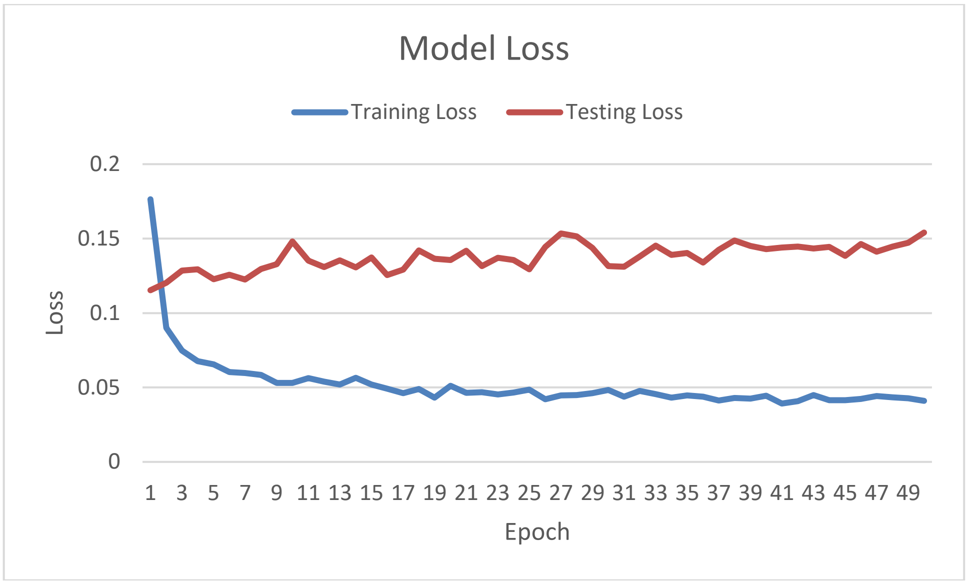


**Fig. 7:** Model accuracy and loss in each epoch before early stopping were used which shows that the model was overfitted before ten epochs were completed

## 5. Results and discussion

The results obtained during the study are shown in Table 2, based on the performance metrics, as mentioned above. As shown in Table 2, the average accuracy, precision, recall, and F1 are 98.34%, 98.45%, 98.23%, and 98.29%. Further, the model achieved 99.8% average AUC in ROC curve. These metrics' values indicate that the model is well suited for detecting phishing attacks. In order to illustrate the accuracy of the proposed solution in a more precise way, several methods were used with the experimental data source with different feature sets. The result of the experiment is shown in Table 3. The results show that the proposed model is outperformed compare to the other methods with the data source by achieving high prediction accuracy.

**Table 2:** Results of the experiments

| | Accuracy | Precision | Recall | F1 Value | AUC |
|---|---|---|---|---|---|
| Experiment 1 | 98.49% | 98.23% | 98.75% | 98.44% | 99.8% |
| Experiment 2 | 98.46% | 98.74% | 98.18% | 98.41% | 99.8% |
| Experiment 3 | 98.07% | 98.39% | 97.77% | 98.02% | 99.7% |
| Average | 98.34% | 98.45% | 98.23% | 98.29% | 99.8% |

According to the explored literature, this is the first time CNN and LSTM techniques are combined to detect phishing attacks based on both HTML and URL features. Previously, a stack model was introduced by Li et al. (2019) to detect phishing attacks using both HTML and URL features, and it had an accuracy of 97.3%. That is the best model found in the literature to compare the model presented in this paper since both used HTML features and URLs in phishing detection. The model presented here has several advantages over the benchmarked model. The detection accuracy is improved by 1.0%, and it is one advantage. Although both models have the HTML feature extraction process, the presented model is not using any URL feature extraction with the use of expert knowledge, which is another benefit getting over the benchmarked model. The latest approach introduced to the phishing area is the HTMLPhish (Opara et al., 2019). It achieved the detection accuracy of 97.2%, and that accuracy is also low compared to proposed model accuracy. However, HTMLPhish is not using any manual feature extraction. That is a drawback of the proposed solution since it used manual feature extraction from the HTML pages. Although the model used manual HTML feature extraction, incorporating URLs with the solution added some benefits to the model over HTMLPhish to have better accuracy.

**Table 3:** Results of the experiments which were done with different possible methods

| Approach | Feature Set | Accuracy |
|---|---|---|
| LSTM only | only URLs | 88.67% |
| LSTM + 1D Conv | only URLs | 96.20% |
| 1D Conv | only 15 HTML Features | 91.70% |
| Proposed Model | 15 HTML Features and URLs | 97.74% |

Aforementioned in the literature, there are LSTM and CNN based approaches that do not need the feature extraction process to detect phishing attacks. Those are based on only URLs. Although the approaches have high accuracy compared to the proposed model, several challenges exist, as mentioned in the State of the Art section. However, those challenges are not affecting the proposed model since it is a combination of both URLs and HTML features. Although the proposed model takes more detection time as argued by Bahnsen et al. (2017), analyzing both HTML content and URL help to get the accurate decision and reduce the spoofed

URL attempts, which can be produced by smart phishers. Further, Table 3 is a perfect showcase of how well the experimental data source performed with the different types of detection methods, which is possible to have in phishing detection. It indicates that the use of both URL and HTML content analysis is increased the detection accuracy than using only URLs or HTML features.

## 6. Conclusion and future works

In this work, a novel approach to detect phishing attacks was introduced. The solution depends on HTML content and URL of a web site. The URLs were trained in the LSTM network and the 1D convolutional network. The network used URLs as input, and expert knowledge is not required for URL feature extraction. Another 1D convolutional model was used to train HTML features, and the HTML features were extracted using a feature extraction model. Finally, these two networks were trained separately and combined through a sigmoid layer by dropping the last layer of each model to have the proposed classifier. The experiment used a self-constructed data source with 20000 phishing and 20000 legitimate data. The phishing data mainly collected from the PhishTank and phishing web site data source of the University Malaysia Sarawak. Expect for PhishTank data; other collected phishing data were validated either by PhishTank or Google Safe Browsing API to have an accurate phishing data source. Legitimate data was collected through the Google search engine by running a Python script. The experiment used three partitions of the data source as training, testing, and validation. The proportions used in each partition are 70%, 20%, and 10% respectively. The scikit-learn python library is used in data partitioning, and the experiment was done three times to have a less biased model at the end.

The proposed model reached 98.34% in terms of accuracy rate and 99.8% AUC value in the ROC curve. This is the highest accuracy achieved by a phishing detection solution that used both HTML and URLs in the explored literature. Further, the experimental data source was used with few different possible detection methods, and the proposed solution selected as the best by emphasizing both HTML features and URLs is essential in phishing detection. One great advantage incorporates with the solution is eliminating expert interaction for feature extraction in URLs. However, HTML pages are still suffering from expert knowledge, which should be eliminated in the future to have a robust model in phishing detection. Therefore, future studies need to be carried out to overcome that drawback, and if that is a success, then a self-learning model can be implemented to detect phishing attacks without human interaction. Then time to time, the model can do self-learning to update the detection criteria automatically to become a useful model in the rapidly changing nature of phishing. However, the used HTML features do not depend on third-party services. Therefore, real-time applications can be implemented using the proposed model to detect phishing attacks. Several optimization techniques can be used to improve the accuracy, and different HTML feature sets also can be used as future works to check whether the proposed architecture can fine-tune more.

## Compliance with ethical standards

## Conflict of interest

The authors declare that they have no conflict of interest.

## References

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

Afroz S and Greenstadt R (2011). Phishzoo: Detecting phishing websites by looking at them. In the 5th International Conference on Semantic Computing, IEEE, Palo Alto, USA: 368-375. https://doi.org/10.1109/ICSC.2011.52

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

Bahnsen AC, Bohorquez EC, Villegas S, Vargas J, and González FA (2017). Classifying phishing URLs using recurrent neural networks. In the APWG Symposium on Electronic Crime Research, IEEE, Scottsdale, USA: 1-8. https://doi.org/10.1109/ECRIME.2017.7945048

Baslyman M and Chiasson S (2016). "Smells phishy?": An educational game about online phishing scams. In the APWG Symposium on Electronic Crime Research, IEEE, Toronto, Canada: 1-11. https://doi.org/10.1109/ECRIME.2016.7487946

Cao Y, Han W, and Le Y (2008). Anti-phishing based on automated individual white-list. In the 4th ACM Workshop on Digital Identity Management, Association for Computing Machinery, Alexandria, USA: 51-60. https://doi.org/10.1145/1456424.1456434 **PMCid:PMC3451763**

Chen KT, Chen JY, Huang CR, and Chen CS (2009). Fighting phishing with discriminative keypoint features. IEEE Internet Computing, 13(3): 56-63. https://doi.org/10.1109/MIC.2009.59

Chen W, Zhang W, and Su Y (2018). Phishing detection research based on LSTM recurrent neural network. In the International Conference of Pioneering Computer Scientists, Engineers and Educators, Springer, Zhengzhou, China: 638-645. https://doi.org/10.1007/978-981-13-2203-7_52

Chiew KL, Chang EH, Tan CL, Abdullah J, and Yong KSC (2018a). Building standard offline anti-phishing dataset for benchmarking. International Journal of Engineering and Technology, 7(4.31): 7-14.

Chiew KL, Tan CL, Wong K, Yong KS, and Tiong WK (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Information Sciences, 484: 153-166. https://doi.org/10.1016/j.ins.2019.01.064

Chiew KL, Yong KSC, and Tan CL (2018b). A survey of phishing attacks: Their types, vectors and technical approaches. Expert Systems with Applications, 106: 1-20. https://doi.org/10.1016/j.eswa.2018.03.050

Chou N, Ledesma R, Teraguchi Y, Boneh D, and Mitchell JC (2004). Client-side defense against web-based identity theft. Computer Science Department, Stanford University, Stanford, USA.

Dong X, Clark JA, and Jacob J (2008). Modelling user-phishing interaction. In the Conference on Human System Interactions, IEEE, Krakow, Poland: 627-632. https://doi.org/10.1109/HSI.2008.4581513

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

Dunlop M, Groat S, and Shelly D (2010). 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

Egelman S, Cranor LF, and Hong J (2008). You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In the SIGCHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, Florence, Italy: 1065-1074. https://doi.org/10.1145/1357054.1357219

El-Alfy ESM (2017). Detection of phishing websites based on probabilistic neural networks and K-medoids clustering. The Computer Journal, 60(12): 1745-1759. https://doi.org/10.1093/comjnl/bxx035

ENISA (2019). 15 top cyber-threats and trends: ENISA threat landscape report 2018. The European Union Agency for Network and Information Security, Heraklion, Greece. Available online at: https://bit.ly/3g5cVgd

Fu AY, Wenyin L, and Deng X (2006). Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). IEEE Transactions on Dependable and Secure Computing, 3(4): 301-311. https://doi.org/10.1109/TDSC.2006.50

Gu X, Wang H, and Ni T (2013). An efficient approach to detecting phishing web. Journal of Computational Information Systems, 9(14): 5553-5560.

Gupta BB, Tewari A, Jain AK, and Agrawal DP (2017). Fighting against phishing attacks: State of the art and future challenges. Neural Computing and Applications, 28(12): 3629-3654. https://doi.org/10.1007/s00521-016-2275-y

Huang CY, Ma SP, Yeh WL, Lin CY, and Liu CT (2010). Mitigate web phishing using site signatures. In the TENCON 2010-2010 IEEE Region 10 Conference, IEEE, Fukuoka, Japan: 803-808. https://doi.org/10.1109/TENCON.2010.5686582

Jain AK and Gupta BB (2016). A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP Journal on Information Security, 2016: 9. https://doi.org/10.1186/s13635-016-0034-3

Jain AK and Gupta BB (2017). Phishing detection: Analysis of visual similarity based approaches. Security and Communication Networks, 2017: 5421046. https://doi.org/10.1155/2017/5421046

Jain AK and Gupta BB (2018). PHISH-SAFE: URL features-based phishing detection system using machine learning. In: Bokhari M, Agrawal N, and Saini D (Eds.), Cyber security: 467-474. Springer, Singapore, Singapore. https://doi.org/10.1007/978-981-10-8536-9_44

Jeeva SC and Rajsingh EB (2016). Intelligent phishing URL detection using association rule mining. Human-Centric Computing and Information Sciences, 6(1): 1-19. https://doi.org/10.1186/s13673-016-0064-3

Joshi Y, Saklikar S, Das D, and Saha S (2008). Phishguard: A browser plug-in for protection from phishing. In the 2nd International Conference on Internet Multimedia Services Architecture and Applications, IEEE, Bangalore, India: 1-6. https://doi.org/10.1109/IMSAA.2008.4753929 **PMid:18604105**

Khonji M, Iraqi Y, and Jones A (2013). Phishing detection: A literature survey. IEEE Communications Surveys and Tutorials, 15(4): 2091-2121. https://doi.org/10.1109/SURV.2013.032213.00009

LeCun Y, Bengio Y, and Hinton G (2015). Deep learning. Nature, 521: 436-444. https://doi.org/10.1038/nature14539 **PMid:26017442**

Li Y, Yang Z, Chen X, Yuan H, and Liu W (2019). A stacking model using URL and HTML features for phishing webpage detection. Future Generation Computer Systems, 94: 27-39. https://doi.org/10.1016/j.future.2018.11.004

Mao J, Tian W, Li P, Wei T, and Liang Z (2017). Phishing-alarm: Robust and efficient phishing detection via page component similarity. IEEE Access, 5: 17020-17030. https://doi.org/10.1109/ACCESS.2017.2743528

Mohammad RM, Thabtah F, and McCluskey L (2014a). Intelligent rule-based phishing websites classification. IET Information Security, 8(3): 153-160. https://doi.org/10.1049/iet-ifs.2013.0202

Mohammad RM, Thabtah F, and McCluskey L (2014b). Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 25(2): 443-458. https://doi.org/10.1007/s00521-013-1490-z

Nguyen LAT, To BL, Nguyen HK, and Nguyen MH (2014a). An efficient approach for phishing detection using single-layer neural network. In the International Conference on Advanced Technologies for Communications, IEEE, Hanoi, Vietnam: 435-440. https://doi.org/10.1109/ATC.2014.7043427

Nguyen LD, Le DN, and Vinh LT (2014b). Detecting phishing web pages based on DOM-tree structure and graph matching algorithm. In the 5th Symposium on Information and Communication Technology, Association for Computing Machinery, Hanoi, Viet Nam: 280-285. https://doi.org/10.1145/2676585.2676596

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

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

Pham TTT, Hoang VN, and Ha TN (2018). Exploring efficiency of character-level convolution neuron network and long short term memory on malicious URL detection. In the 7th International Conference on Network, Communication and Computing, Association for Computing Machinery, Taipei City, Taiwan: 82-86. https://doi.org/10.1145/3301326.3301336 **PMCid:PMC6162070**

Prakash P, Kumar M, Kompella RR, and Gupta M (2010). Phishnet: Predictive blacklisting to detect phishing attacks. In the IEEE INFOCOM, IEEE, San Diego, USA: 1-5. https://doi.org/10.1109/INFCOM.2010.5462216

Pratiwi ME, Lorosae TA, and Wibowo FW (2018). Phishing site detection analysis using artificial neural network. Journal of Physics: Conference Series, 1140: 1. https://doi.org/10.1088/1742-6596/1140/1/012048

Rosiello AP, Kirda E, and Ferrandi F (2007). A layout-similarity-based approach for detecting phishing pages. In the 3rd International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007, IEEE, Nice, France: 454-463. https://doi.org/10.1109/SECCOM.2007.4550367

Sahingoz OK, Buber E, Demir O, and Diri B (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications, 117: 345-357. https://doi.org/10.1016/j.eswa.2018.09.029

Sahoo D, Liu C, and Hoi SC (2017). Malicious URL detection using machine learning: A survey. Available online at: https://bit.ly/2LKpSyg

Sheng S, Magnien B, Kumaraguru P, Acquisti A, Cranor LF, Hong J, and Nunge E (2007). Anti-phishing phil: The design and evaluation of a game that teaches people not to fall for phish. In the 3rd Symposium on Usable Privacy and Security, Association for Computing Machinery, Pittsburgh, USA: 88-99. https://doi.org/10.1145/1280680.1280692 **PMid:17433705**

Subasi A, Molah E, Almkallawi F, and Chaudhery TJ (2017). Intelligent phishing website detection using random forest classifier. In the International Conference on Electrical and Computing Technologies and Applications, IEEE, Ras Al Khaimah, UAE: 1-5. https://doi.org/10.1109/ICECTA.2017.8252051

Weider DY, Nargundkar S, and Tiruthani N (2008). A phishing vulnerability analysis of web based systems. In the IEEE Symposium on Computers and Communications, IEEE, Marrakech, Morocco: 326-331. https://doi.org/10.1109/ISCC.2008.4625681

Whittaker C, Ryner B, and Nazif M (2010). Large-scale automatic classification of phishing pages. Available online at: https://bit.ly/2WJqZV4

Wu M, Miller RC, and Garfinkel SL (2006). Do security toolbars actually prevent phishing attacks? In the SIGCHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, Montral, Canada: 601-610. https://doi.org/10.1145/1124772.1124863

Zhang Y, Hong JI, and Cranor LF (2007). Cantina: A content-based approach to detecting phishing web sites. In the 16th International Conference on World Wide Web, Association for Computing Machinery, Banff, Canada: 639-648. https://doi.org/10.1145/1242572.1242659