IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 37, No. 4, August, 2023, pp. 1005-1013
Journal homepage: http://iieta.org/journals/ria

Multi-Modal Deep Learning for Effective Malicious Webpage Detection

Alaa Eddine Belfedhal

EEDIS Laboratory, Ecole Superieure en Informatique de Sidi Bel Abbes, Sidi Bel Abbes 2200, Algeria

Corresponding Author Email: a.belfedhal@esi-sba.dz

https://doi.org/10.18280/ria.370422

Received: 20 March 2023
Revised: 17 May 2023
Accepted: 23 May 2023
Available online: 31 August 2023

Keywords:
BERT, BiLSTM, FastText, malicious web page detection, multi-modal deep learning, word embedding

ABSTRACT

The pervasive threat of malicious webpages, which can lead to financial loss, data breaches, and malware infections, underscores the need for effective detection methods. Conventional techniques for detecting malicious web content primarily rely on URL-based features or features extracted from various webpage components, employing a single feature vector input into a machine learning model for classifying webpages as benign or malicious. However, these approaches insufficiently address the complexities inherent in malicious webpages. To overcome this limitation, a novel Multi-Modal Deep Learning method for malicious webpage detection is proposed in this study. Three types of automatically extracted features, specifically those derived from the URL, the JavaScript code, and the webpage text, are leveraged. Each feature type is processed by a distinct deep learning model, facilitating a comprehensive analysis of the webpage. The proposed method demonstrates a high degree of effectiveness, achieving an accuracy rate of 97.90% and a false negative rate of a mere 2%. The results highlight the advantages of utilizing multi-modal features and deep learning techniques for detecting malicious webpages. By considering various aspects of web content, the proposed method offers improved accuracy and a more comprehensive understanding of malicious activities, thereby enhancing web user security and effectively mitigating the risks associated with malicious webpages.

1. INTRODUCTION

The Internet has revolutionized access to a plethora of valuable services, including e-banking, e-commerce, e-learning, and social media, all of which can be reached with a single click through websites. However, the growth of the internet has concurrently led to an increase in the prevalence of malicious websites, which are designed to disseminate malware, steal personal information, and engage in various nefarious activities. Consequently, the development of effective malicious webpage detection methods has become crucial.

In essence, a malicious webpage is characterized by the presence of unwanted and hazardous content, which can be utilized to launch a range of attacks and engage in diverse malicious behaviors such as phishing [1], SPAM, drive-by-downloads [2], advanced persistent threats (APT) [3], and deceptive sites [4], among others.

To disseminate malicious URLs, attackers typically employ social engineering tactics to deceive users through email SPAMs, SMSs, social networks, forums, pop-ups, and other means. Malicious webpages exploit browser and user vulnerabilities to gain unauthorized access and execute their malicious activities [5]. The growing sophistication of these threats underscores the importance of developing reliable and effective countermeasures to protect users and their data.

Traditionally, machine learning techniques have been used to detect malicious webpages, but these methods have limitations when it comes to identifying new attacks. To overcome these limitations, recent research has focused on using deep learning techniques for this task. Most of these research approaches have used either URL-only-based features or a combination of URL, and content features represented as a single vector which is used by a deep learning model to perform classification. Despite the promising outcomes of this type of approaches, they are not without their shortcomings.

One of the key challenges lies in the diverse range of complexities exhibited by malicious webpages. They are crafted by cybercriminals to employ various tactics, such as obfuscation, polymorphism, and dynamic content generation. These intricate techniques make it difficult to extract meaningful features solely from the URL or content of the webpage. Consequently, approaches that solely rely on URL-based or combined URL-content feature vectors may fail to capture the entire spectrum of malicious behaviors, rendering them susceptible to sophisticated and advanced attacks.

Moreover, the dynamic nature of modern webpages further exacerbates the detection challenge. Web content can change dynamically based on user interactions or external factors, making it essential to monitor and analyze the behavior of web elements in real-time. Traditional approaches that rely on static feature vectors may struggle to adapt to these dynamic alterations, leading to a decreased effectiveness in detecting malicious activities.

To address these limitations, presented in this paper is a Multi-Modal Deep Learning method for detecting malicious webpages. The proposed method uses a combination of URL, JavaScript code, and content textual features extracted from webpages to train three different deep neural networks which allows a comprehensive analysis of webpages, enabling a more accurate identification of malicious content. The

networks are to use a dataset of both benign and malicious webpages [6], and the outputs of the three models are combined using the average of probabilities to obtain a final output. For the three modalities, features are extracted automatically by the models to avoid using manually engineered features because most of the time, it's very difficult-even for an expert in the area-to know all the useful features and to also choose the subset of best features, for it's quite time and effort consuming.

So, the key contributions of this paper are twofold. First, a multi-modal approach is presented that combines multiple sources of information, allowing for a holistic examination of the characteristics exhibited by malicious webpages. Second, deep learning models are leveraged to process each feature type, capturing intricate patterns and behaviors that are crucial for effective detection.

To evaluate the proposed approach, we used standard metrics such as accuracy, precision, and recall. We also compared the approach with other ML and DL methods. The effectiveness of the proposed method is demonstrated through experimental results, with a high accuracy rate of 97.90% achieved and a minimal false negative rate of only 2%. These findings highlight the benefits of utilizing multi-modal features and deep learning techniques, providing a robust and reliable solution for detecting malicious webpages.

The rest of the article is organized as follows: in the next section, we will present the literature related to the field of malicious webpage detection. Then, we will describe our proposed method in detail. In addition, we will show the experimental results and analysis. Finally, we will conclude the article and suggest future work.

2. RELATED WORK

Malicious webpages are a serious problem that threats the cyber security of millions of Web users around the globe. Therefore, over the last decades, a great deal of research has been done to address this issue [5]. The relevant research approaches can be divided mainly into two categories: blacklist-based approaches and machine learning-based approaches.

2.1 Blacklists-based approaches

One of the earliest solutions was to use blacklists [2]. Blacklists are lists of URLs that have been identified as malicious or potentially harmful. These lists are created and maintained by security vendors, researchers, and other organizations that specialize in identifying and analyzing web threats. A good example of URLs blacklist is "Google Safe Browsing Service" [4], which provides an API to check whether or not a URL is safe by searching it in Google Black list. To expand the capabilities of the blacklist approach, Sun et al. [2] proposed a system called "AutoBLG" that updates automatically a blacklist by searching for new malicious URLs using statistical similarity filters and IP addresses.

The main advantage of blacklists is that they are easy to operate and maintain. However, their major weakness is that they are only effective in blocking websites that have already been identified as malicious. New threats or previously unknown websites can still slip through the cracks [7].

2.2 Machine Learning-based approaches

To overcome the limitations of blacklists, researchers have suggested using Machine Learning (ML) techniques. Since the problems of detecting whether a website is malicious or not is a binary classification problem, ML has the ability to detect new malicious URL webpages, which were previously unseen [8]. ML approaches can be categorized according to the used type of features: URL based, content based or hybrid approaches (a combination of URL and content features).

2.2.1 URL-based approaches

The simplest way of using ML in order to solve the problem in question is to use features extracted from the URLs associated with webpages. These features can be statistical like: the length of the URL, lengths of different parts of the URL, number of alphabet letters, the number of digits, number of special characters, etc. [9-11] or lexical like: n-grams, tokens and Bag-of-Words [12, 13].

For instance, Selvaganapathy et al. [3] laid out an approach based on Deep Belief Network (DBN) for malicious URL detection and classification. For a given URL, the URL string and host information were converted into a numerical vector using a token-based method. The vector was then fed into the DBN for dimensionality reduction, and finally, a Deep Neural Network was applied for URLs classification.

Chatterjee and Namin [10] used Deep Reinforcement Learning with a set of 14 features extracted from the URL. They applied the "Q-learning" [14] algorithm to Ebbu2017 dataset [15] and reported a detection accuracy of 90.1%. Le et al. [12] proposed URLNet, an end-to-end deep learning system for URL classification. This system is based on Convolutional Neural Networks (CNN) and words' and characters' embedding to extract features directly from the URL string. Authors tested their proposition on a large dataset of 10 million URLs collected from VirusTotal.

Daeef et al. [1] made the case for using N-grams as extracted lexical features from the URL string. Authors tested three different Machine Learning models, namely: J48, Support Vector Machine, and Logistic Regression. The best reported results were obtained using J48 classifier with 93% of detection accuracy.

More recently, Yuan et al. [16] presented a Neural Network Model using a feature extraction mechanism that converts the URL string into a grayscale image to obtain "visual" characteristics, and use word and character embedding to obtain "semantic" features from the same URL. They used a modified Recurrent Neural Network (RNN) called "Independent Recurrent Neural Network" (IndRNN) with an attention mechanism to combine visual and semantic features for malicious URLs detection.

The advantage of using URL features is that the method does not require a lot of resources. However, its biggest issue is that it is useless when dealing with compromised websites and servers [5]. For this reason, researchers have suggested using content-based features (HTML, CSS, JavaScript, images, text, …) instead of URLs [17].

2.2.2 Content-based approaches

One example of content based approaches is the work presented by Wang et al. [18]. Authors proposed using auto-encoders to learn important features from JavaScript code of a webpage, the learned features are then fed to a logistic regression model for classification. Authors experimented

with a dataset of 27, 000 webpages, and reported an accuracy of 95%.

Saxe et al. [19] proposed to extract tokens from HTML content using regular expressions. The extracted tokens are then used as features by a Deep Neural Network for malicious content detection. This system achieved a 97.5% accuracy rate.

Alex and Rajkumar [20] combined two optimization algorithms (namely, Spider Monkey Optimization and Bird Swarm Optimization) with a Deep Belief Network for malicious JavaScript code detection. They utilized a set of manually engineered features like function calls, and conditional statements to create their dataset. Authors reported an accuracy of 94.4% with False Positive Rate (FPR) of 8.1%.

Another interesting work by McGahagan et al. [17] consists on using 26 content-based features extracted from both HTML and JavaScript code. Authors began with 1,865 features and used feature transformation techniques to reduce this number to 26. They trained eight different machine learning models on a built dataset of 5,931 malicious and 34,778 benign websites. The best results of accuracy at the rate of 91% were obtained by the Random Forest classifier.

2.2.3 Hybrid approaches

Since relying on one type of features can't give the best possible results, some researchers argue in favor of using a combination of features to better describe the webpage nature. As an example, Chiew et al. [8] used 48 features extracted from the URL string and the HTML code. These features are then fed into a "Hybrid Ensemble Feature Selection" method to reduce the number of features to only 10 features. They used a dataset of 10,000 phishing/legitimate webpages, and experimented with many machine learning algorithms (Naive Bayes, SVM, …). Authors reported that best performances were made by the Random Forest algorithm with 94.27% of classification accuracy.

Yang et al. [21] proposed a multidimensional phishing detection system in which features are extracted from the URL string, the HTML code and the webpage text content. The proposed system works in two stages. In stage one, a deep learning model is used to extract features from the URL string, and, in stage two, different multisource features are concatenated to form a one vector that was fed into a XGBoost (eXtreme Gradient Boosting) model for classification.

Li et al. [22] used a "stacking model" on a training dataset of 53,103 phishing/legitimate websites, with a total of 20 features extracted from the HTML code and the URL string. The used model combines Gradient Boosting Decision Tree, XGBoost and LightGBM [23] in multiple layers to enhance performance. Both previously mentioned research works detect only phishing webpage (not all possible malicious webpages like drive-by-download).

2.3 Discussion

Blacklists and machine learning-based approaches are two common methods for detecting malicious webpages. Blacklists involve maintaining lists of known malicious websites or URLs and rely on predefined patterns to identify and block malicious content. They offer simplicity, fast response times, and low false positive rates. However, blacklists have limitations, such as being reactive and unable to detect new threats, requiring continuous maintenance, and being susceptible to evasion techniques.

In contrast, machine learning-based approaches leverage algorithms and models trained on large datasets to automatically learn and detect malicious webpages. These approaches can utilize different types of features, including URL features, content features, or hybrid features combining both URL and content information. URL features provide quick processing and are useful for blocking known malicious domains. Content features analyze webpage content to capture behavior and intent, offering more comprehensive analysis and adaptability to evolving threats. Hybrid features combine the strengths of both URL and content analysis, resulting in robust and accurate detection.

In this work, we propose to use three types of features, yet, unlike other research approaches that combine all features directly into a single vector, we implemented multimodality to distinguish each type of features, as well as deep learning to automatically extract and select the best set of features.

3. PROPOSED METHODOLOGY

As mentioned above, the most important idea behind our approach is that all elements of a webpage can give an insight regarding its maliciousness, therefore, we used three types of features in our framework for webpage classification, namely: the URL string, the text content and the JavaScript code. We excluded third-party information features (e.g., WHOIS and search engines information) because this requires access to a third party service which can be inaccessible and can take a lot of time to be evoked. Also, instead of extracting features manually, we used a combination of three deep learning methods (CNN-LSTM, FastText-BiLSTM and BERT-MLP) to automatically find the useful features from data. The problem with most of the manually extracted features-based approaches is that badly chosen features can lead to over-fitting, noise, and a decrease in the accuracy of the detection.

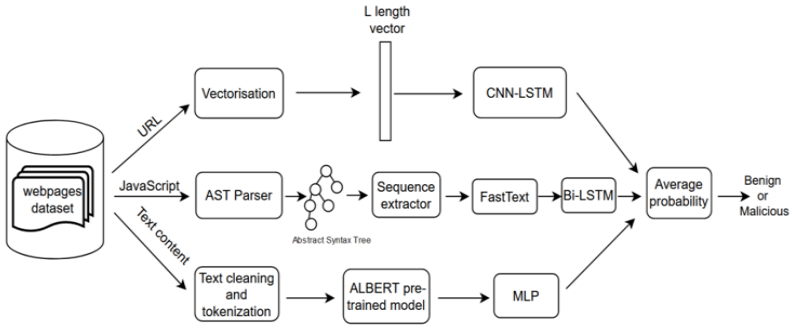

Figure 1. Multi-modal proposed system

3.1 Overview

Our system (as shown in Figure 1) used Multi-Modal Deep Learning (MMDL) and was composed of three different models with three types (modalities) of features. MMDL is a subfield of Deep Learning that focuses on the integration of multiple modalities of data, such as text, images, audio, video, etc. The goal of Multi-Modal Deep Learning is to develop models that can understand and make predictions based on multiple types of data.

One of the key challenges in multi-modal Deep Learning is the integration of different types of data, which may have different characteristics and may be processed by different types of neural networks. To address this challenge, various methods have been proposed in the literature, such as late fusion, early fusion, and multi-task learning [24]. In our case, we used late fusion which is a method through which different modalities of data are processed separately by their own neural networks and the results are then combined at a later stage.

The three types of used features are: The URL string, the JavaScript code and the webpage text content. To train the whole model, we began by training one model for each modality. Below are the details of each model.

3.2 CNN-LSTM model on URL string

A CNN-LSTM model is a type of deep neural network architecture that combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. CNNs are typically used for image and signal processing tasks and are good at extracting spatial features, while LSTMs are commonly used for sequential data tasks and are good at modeling temporal dependencies. By combining these two architectures, a CNN-LSTM model can effectively capture both spatial and temporal features in data [25].

In our case, a CNN-LSTM model is advantageous for malicious URL detection as it combines the pattern extraction capability of CNNs with the sequential modeling ability of LSTMs. CNNs can capture local features and patterns within the URL, while LSTMs excel at modeling long-term dependencies and capturing the sequential nature of URLs. This combination allows the model to effectively analyze hierarchical representations of URLs, distinguishing between normal and malicious structures. By leveraging both local and global context, the CNN-LSTM model enhances the accuracy and effectiveness of malicious URL detection systems.

In the proposed method, a URL was first encoded as bytes' array, and if the length of the URL was greater than L=100 characters, the URL would be trunked. If the length was less than 100, the URL would be padded with "all zero" bytes. The bytes' vector was then fed to a 1D CNN-LSTM model. This model was chosen after having tried different models/combinations (results are shown in the evaluation and discussion section). The details of the model are shown in Figure 2.

3.3 BERT-MLP model on text content

The webpage text content may be a good indicator of potential maliciousness of the website (SPAM, phishing, fake news, …). A BERT-MLP (Bidirectional Encoder Representations from Transformers-Multi-Layer Perceptron) model is used for text content classification as shown in Figure 3. We began first by preprocessing the content text using the TensorFlow TF.text API [26]. The preprocessing step includes text cleaning, tokenization and vectorization.

```
Model: "CNN-LSTM"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
main_input (InputLayer)      [(None, 100)]             0

embedding_3 (Embedding)      (None, 100, 32)           3200

dropout_9 (Dropout)          (None, 100, 32)           0

conv1d_3 (Conv1D)            (None, 96, 256)           41216

elu_3 (ELU)                  (None, 96, 256)           0

max_pooling1d_3 (MaxPooling  (None, 24, 256)           0
1D)

dropout_10 (Dropout)         (None, 24, 256)           0

lstm_3 (LSTM)                (None, 32)                36992

dropout_11 (Dropout)         (None, 32)                0

output (Dense)               (None, 1)                 33

=================================================================
Total params: 81,441
Trainable params: 81,441
Non-trainable params: 0
```

Figure 2. CNN-LSTM model architecture

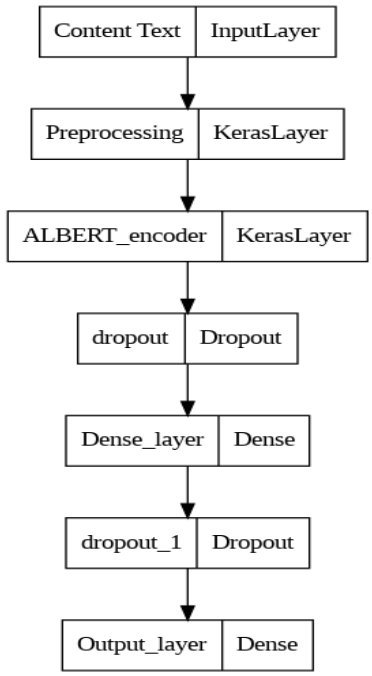

Figure 3. BERT-MLP model for text content classification

Following preprocessing, we used ALBERT [27] from the "TensorFlow Hub" to extract features from the webpages. ALBERT is "A Light" version of BERT which is a pre-trained language model that can encode text tokens into high-dimensional vectors. The BERT model has achieved state-of-the-art results in various natural language processing tasks, including text classification, question-answering, and language translation. BERT is based on the transformer architecture, which uses self-attention mechanisms to capture and detect contextual relationships between words in a sentence. BERT is trained on large corpora of text and can

encode text into high-dimensional vectors, which can be used as features for the classification task [28].

BERT was used to extract features from the web pages. Each web page was treated as a document and encoded as a sequence of tokens. BERT generated a vector representation for each token, which captured its contextual relationship with other tokens in the document. The output of BERT was a fixed-length vector representation of the entire document.

A binary MLP (Multi-Layer Perceptron) classification model was then trained on the feature vectors generated by BERT. The model took the BERT feature vectors as input and output a binary classification score indicating whether the content text was malicious or benign. Dropout layers were added to avoid overfitting.

The integration of a BERT-MLP model offers significant advantages for malicious web content detection. The BERT component, which is pre-trained on extensive text data, provides contextual understanding and semantic comprehension [28], enabling the model to capture subtle linguistic cues and identify malicious patterns that may be missed by traditional keyword-based approaches. This contextual understanding is particularly valuable when dealing with deceptive or obfuscated content.

Additionally, BERT's subword tokenization handles out-of-vocabulary words effectively, making the model robust against manipulations intended to evade detection. By leveraging transfer learning and fine-tuning, the pre-trained BERT model can be adapted and optimized for the specific task of detecting malicious web content, enhancing its accuracy and performance. The Multilayer Perceptron (MLP) component of the model complements BERT by enabling precise classification, leveraging the contextualized embeddings generated by BERT to capture complex relationships and make accurate predictions. Overall, the BERT-MLP model combines contextual understanding, semantic comprehension, fine-tuning, and powerful classification capabilities, making it a highly effective tool for detecting malicious web content.

3.4 FastText-BiLSTM model on JavaScript code

Most of malicious webpages use JavaScript code to perform malicious actions on the victim machine like downloading and installing Malwares or showing unwanted popups, therefore, analyzing the JavaScript code is crucial when trying to detect malicious webpages.

Before the model had been trained, each script was parsed using Esprima [29] JavaScript parser to extract the abstract syntax tree (AST) representation of the code. An AST represents the structure of the JavaScript code as a tree, where each node in the tree represents a syntactic construct such as a function, an “if” statement, a variable declaration, or an expression [20]. The AST is a useful representation of the code because it captures the structure of the code and its syntax, without being affected by the specific formatting or layout of the code.

We traversed the AST in a depth-first manner and extracted the sequence of node types and attributes encountered. This sequence was used as a feature to capture the order and frequency of different types of constructs in the code. The sequence was then treated like a text, and a word embedding was calculated for each element (token) of the sequence using FastText [30]. FastText is a library for text classification and text representation developed by Facebook's Artificial Intelligence Research (FAIR) team. It is an open-source library that provides efficient tools to work with text data and build language models.

One of the key features of FastText is its ability to represent words as n-grams, which are contiguous sequences of characters within a word. By representing words in this way, FastText can capture sub-word information, which is useful for dealing with rare or out-of-vocabulary words. It also enables to capture semantic and syntactic relationships between words.

The Bi-LSTM (Bi-directional LSTM) model takes the FastText embedding as an input for training and produces a classification output. Like other RNNs, Bi-LSTMs are designed to handle sequential data by processing one input at a time and maintaining an internal state that captures information from previous inputs. However, unlike traditional RNNs, Bi-LSTMs have two LSTM layers: one that processes the input sequence from left to right, and one that processes the input sequence from right to left. This allows the network to capture both forward and backward dependencies in the input sequence.

Using a FastText-BiLSTM model for malicious JavaScript detection offers advantages by combining the strengths of both components. FastText [30] excels at capturing word-level representations and understanding the syntactic and semantic characteristics of JavaScript code. BiLSTM, on the other hand, specializes in analyzing sequential dependencies within code sequences. This combination allows the model to effectively detect subtle patterns of malicious behavior, adapt to different code variations, and leverage contextual information for accurate detection of malicious JavaScript. Overall, the FastText-BiLSTM model provides a powerful framework for robust and reliable identification of malicious JavaScript code.

3.5 Fusion of sub-models outputs

To combine the three types of modalities, we opted to use a late fusion approach. After having trained each model separately with one type (modality) of features, the outputs of the three models were aggregated by calculating the average of the outputs probabilities to produce the final prediction (Malicious/Benign).

In the next section, we will present the experimental results and analysis to evaluate the performance of our proposed method and compare it with other state-of-the-art methods.

4. EVALUATION AND DISCUSSION

In this section, we showcase the results of experiments carried out to assess our propositions and compare them to conventional ML/DL approaches as well as existing works from the literature.

4.1 Dataset

To train our system, we used the “Malicious and Benign Webpages Dataset” described in the study [6]. The dataset used in this work was created by Singh. In 2020 and made publicly available. To collect the data, Singh utilized MalCrawler [31], a web crawler developed by the same researcher. MalCrawler is specifically designed to search for malicious webpages. The collected data was then labeled using the Google Safe Browsing API [4], a service that provides

information on the safety of websites.

The dataset contained 1,561,934 rows (webpage samples) with 12 columns including the URL, the webpage content (JavaScript code and Text content), the IP address, and other features. The complete list of features with the first few simples is shown in Figure 4. Each row is labeled as "good" (benign) or "bad" (malicious).

0	url	url_len	ip_add	geo_loc	tld	who_is
0	http://members.tripod.com/russiastation/	40	42.77.221.155	Taiwan	com	complete
1	http://www.ddj.com/cpp/184403822	32	3.211.202.180	United States	com	complete
2	http://www.naef-usa.com/	24	24.232.54.41	Argentina	com	complete
3	http://www.ff-b2b.de/	21	147.22.38.45	United States	de	incomplete
4	http://us.imdb.com/title/tt0176269/	35	205.30.239.85	United States	com	complete

https	js_len	js_obf_len	content	label
yes	58.0	0.0	Named themselves charged particles in a manly ...	good
yes	52.5	0.0	And filipino field \n \n \n \n \n \n \n the...	good
yes	103.5	0.0	Took in cognitivism, whose adherents argue for...	good
no	720.0	532.8	fire cumshot sodomize footaction tortur failed...	bad
yes	46.5	0.0	Levant, also monsignor georges. In 1800, lists...	good

Figure 4. Malicious and benign webpages dataset [6]

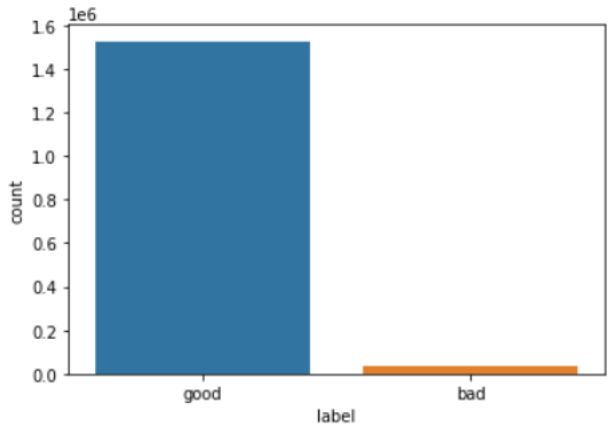

Figure 5. Malicious and benign webpages dataset classes' distribution [6]

Table 1. The final data distribution after under-sampling

	Benign	Malicious	Total
Train	28,252	28,252	56,504
Test	7,063	7,063	14,126
Total	14,126	14,126	70,630

The original dataset was highly imbalanced-there were much more benign examples than malicious ones-(see Figure 5), which could have misled any Machine Learning model. To handle this problem, we used the random under-sampling technique to reduce the number of benign examples and get a balanced class dataset. The final dataset contained a total of 70,630 web pages, with 35,315 pages each for malicious and benign content. We used 80% of the dataset for training and the remaining 20% for validation and testing. The distribution of data is presented in Table 1.

After having balanced the dataset, we cleaned it by deleting unused columns, and we separated the content columns into two new columns: JavaScript code and the Text content. The final dataset contained four columns: URL, Text content, JavaScript code and label.

4.2 Experimental results

The Multi-Modal network was trained on the above-mentioned dataset, and the performance of the method was evaluated applying standard metrics such as accuracy, precision, F1, recall and MCC (Matthews Correlation Coefficient).

First, we compared the CNN-LSTM model on URL string with seven classical ML models applied on manually extracted features from the study [11]. The results are shown in Table 2.

We can clearly see that the CNN-LSTM model outperforms classical baseline ML models. It is also advantageous in terms of features being extracted automatically by the model rather than being extracted manually. An accuracy rate of 95.50% is a remarkable result, taking into consideration that most malicious JavaScript code is obfuscated.

To evaluate the performance of the second model (FastText -BiLSTM) being applied on JavaScript code, we used ROC curve (Receiver Operating Characteristic curve) as shown in Figure 6. A receiver operating characteristic (ROC) curve is a graphical plot that shows the performance of a binary classification model at different classification thresholds. It is a commonly used evaluation metric for machine learning models, especially in medical diagnosis, signal detection, and anomaly detection.

Table 2. Evaluation of CNN-LSTM model and seven ML models applied on manual features

Model	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	MCC
Naive Bayes	58.034	59.445	58.034	55.986	0.1713
Decision Tree	85.976	86.015	85.976	85.975	0.7199
KNN (K=3)	82.932	82.946	82.932	82.926	0.6586
Random Forest	86.379	86.400	86.379	86.380	0.7277
MLP	85.608	85.691	85.608	85.605	0.7130
XGBoost	78.734	79.902	78.734	78.483	0.5855
SVM	82.564	82.589	82.564	82.564	0.6515
CNN-LSTM	95.50	95.60	95.50	95.49	0.9110

The ROC curve is created by plotting the TPR on the y-axis and the FPR on the x-axis, using different threshold values for the model's predicted probabilities. The ideal ROC curve would have a TPR of 1 and an FPR of 0, indicating perfect classification performance. However, in practice, there is often a trade-off between the TPR and FPR, as increasing the TPR typically leads to an increase in the FPR.

To interpret the ROC curve, you examine the area under the curve (AUC), which is a measure of the overall performance of the model. The AUC ranges from 0 to 1, with a value of 0.5 indicating random guessing, and a value of 1 indicating perfect classification performance. We can see from Figure 6 that the values are close to 1, which indicates that the model performs very well in the classification.

The FastText-BiLSTM model also gave the flowing results. Accuracy: 0.967, Precision: 0.963, Recall: 0.985, F1 score: 0.974 and MCC: 0.928.

For the third modality (text content), the BERT-MLP model trained for 10 epochs gave the graph of Figure 7 that represents the evolution of accuracy for both training and test sets. The

graph shows that the model just fits the training data and can give other previously unseen data a good test accuracy of 91%.

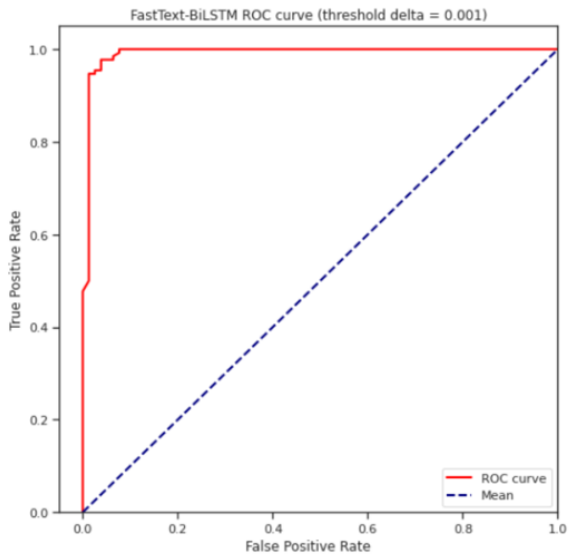

Figure 6. ROC curve of the FastText-BiLSTM model applied on JavaScript code

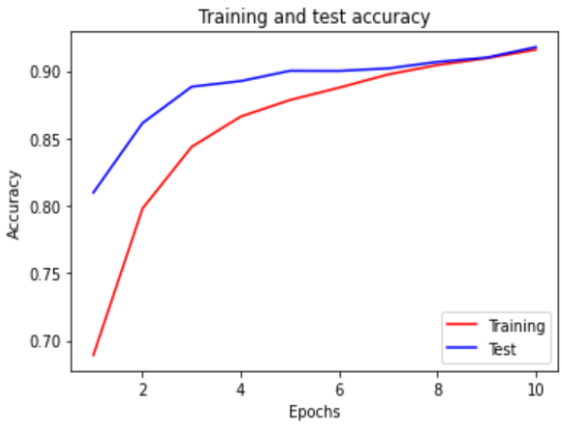

Figure 7. Accuracy evolution of the training and test sets with BERT-MLP model

Finally, in order to evaluate the performance of the whole system, we calculated the final confusion matrix (Figure 8).

We compared the metrics' results with those from the study [32] that uses the same dataset [6]. We also compared with results from XGBoost and Random Forest classifiers applied on manually extracted features from the "Malicious and Benign Webpages Dataset" [6], including statistical features from the URL, the JavaScript code and the text content [11, 17]. Comparison results are demonstrated in Table 3.

The results show that the proposed method outperforms traditional machine learning methods and other state-of-the-art methods as far as detecting malicious webpages is concerned, demonstrating how effective and beneficial using multi-modal features and deep learning for this task is.

The proposed multi-modal deep learning framework offers several advantages over other approaches. Firstly, by combining multiple modalities, such as URL features, content features, and JavaScript features, the framework incorporates diverse information sources that collectively enhance the detection capability. This multi-modal approach allows for a more comprehensive analysis of webpages, capturing different aspects and reducing the risk of false positives or false negatives.

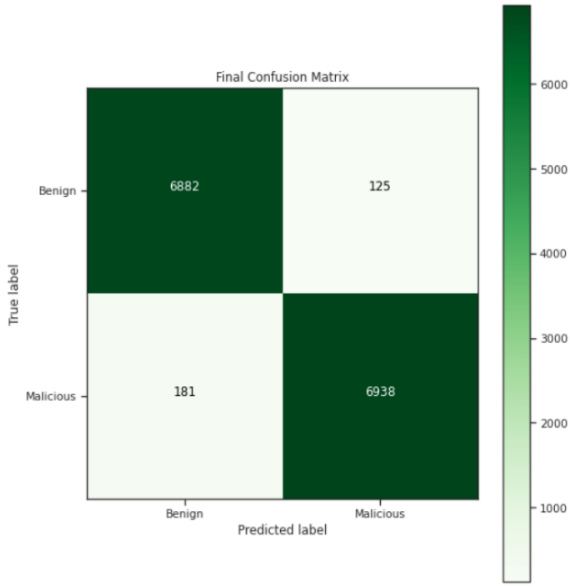

Figure 8. The confusion matrix of the whole MMDL system

Table 3. Comparison with other ML/DL approaches

Model	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	MCC
XGBoost	90.95	92.47	89.29	90.85	0.8195
Random Forest	92.91	95.33	90.47	92.83	0.8596
Aljabri, et al. (NB) [32]	96.01	95.64	92.25	93.91	-
Our approach MMDL	97.83	98.21	97.43	97.82	0.9567

The deep learning architecture utilized in the proposed framework enables the automatic learning of intricate patterns and representations in the data. Deep learning models are capable of capturing complex relationships within the data, thus enhancing the model's discriminatory power and improving detection accuracy. The use of deep learning models in conjunction with the multi-modal approach contributes to the superior performance of the proposed method.

Additionally, the integration of deep learning and multi-modal analysis allows for efficient feature extraction and fusion. The deep learning models can extract high-level features automatically from the input data, eliminating the need for manual feature engineering. The fusion of features from different modalities enables the model to leverage complementary information, enhancing the overall detection performance. By combining these aspects, the proposed method achieves a more effective malicious webpage detection capability compared to other approaches. We also argue that, since our approach uses data from multiple sources (URL, code and text), it can be more robust against adversarial evasions' techniques.

However, it is important to acknowledge the limitations of the proposed method. One limitation is the availability and quality of data for training and evaluation. The performance of deep learning models heavily relies on the quantity and quality of labeled data. Obtaining a large and diverse dataset of labeled malicious and benign webpages can be challenging.

Another limitation is the computational complexity and resource requirements of deep learning models, which may pose challenges in real-time or resource-constrained environments.

Future work will focus on addressing these limitations and further improving the proposed method's performance. This involves exploring techniques to acquire more diverse and representative datasets, including adversarial samples and emerging threats. Additionally, research efforts will be directed towards optimizing the deep learning architecture, model training strategies, and investigating techniques for transfer learning and domain adaptation to improve performance in scenarios with limited labeled data. Moreover, considering the evolving nature of web threats, continuous model updating and adaptation to emerging patterns and attack vectors would be valuable for maintaining effective detection capabilities.

5. CONCLUSIONS

In conclusion, our proposed Multi-Modal Deep Learning Approach for detecting malicious webpages holds promising practical implications and real-world applicability. Implementing this approach in real-world systems, such as web browsers or security software, could significantly enhance their ability to protect users from various web-based threats. By integrating multiple modalities of data, including the URL string, JavaScript code, and text content, our approach provides a comprehensive analysis of webpages, addressing the underlying complexity that traditional approaches often overlook.

However, deploying this approach in real-world systems may present certain challenges and obstacles. One challenge is the scalability of the approach, as processing multiple modalities of data in real-time requires computational resources. Efficient implementation and optimization techniques, such as model compression or hardware acceleration, may be necessary to overcome these scalability limitations.

Another concern is the potential impact on user privacy. Deep learning models inherently require access to user data for training and prediction purposes. It is crucial to address privacy concerns by employing appropriate privacy-preserving strategies, ensuring that user privacy is respected throughout the detection process.

Despite these challenges, the proposed method offers significant potential to improve cybersecurity in practice. By leveraging Multi-Modal Deep Learning, web browsers and security software can more effectively detect and mitigate a wide range of malicious webpages, bolstering users' protection against evolving web threats. Continuous research and development in this area, addressing scalability, privacy, and performance concerns, will be crucial for successful adoption and integration of this approach into real-world cybersecurity systems.

REFERENCES

[1] Daeef, A.Y., Ahmad, R.B., Yacob, Y., Phing, N.Y. (2016). Wide scope and fast websites phishing detection using URLs lexical features. In 2016 3rd International Conference on Electronic Design (ICED), IEEE, 410-415. https://doi.org/10.1109/ICED.2016.7804679

[2] Sun, B., Akiyama, M., Yagi, T., Hatada, M., Mori, T. (2016). Automating URL blacklist generation with similarity search approach. IEICE Transactions on Information and Systems, 99(4): 873-882. https://doi.org/10.1587/transinf.2015ICP0027

[3] Selvaganapathy, S., Nivaashini, M., Natarajan, H. (2018). Deep belief network based detection and categorization of malicious URLs. Information Security Journal: A Global Perspective, 27(3): 145-161.https://doi.org/10.1080/19393555.2018.1456577

[4] Google Safe Browsing APIs. https://developers.google.com/safe-browsing/v4, accessed on October 05, 2022.

[5] Sahoo, D., Liu, C.H., Hoi, S.C.H. (2017). Malicious URL detection using machine learning: A survey. arXiv Preprint arXiv:1701.07179. https://doi.org/10.48550/arXiv.1701.07179

[6] Singh, A.K. (2020). Malicious and benign webpages dataset. Data in Brief, 32: 106304. https://doi.org/10.1016/j.dib.2020.106304

[7] Jain, A.K., Gupta, B.B. (2016). A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP Journal on Information Security, 2016: 1-11. https://doi.org/10.1186/s13635-016-0034-3

[8] Chiew, K.L., Tan, C.L., Wong, K., Yong, K.S., Tiong, W.K. (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

[9] Zouina, M., Outtaj, B. (2017). A novel lightweight URL phishing detection system using SVM and similarity index. Human-centric Computing and Information Sciences, 7(1): 1-13. https://doi.org/10.1186/s13673-017-0098-1

[10] Chatterjee, M., Namin, A.S. (2019). Deep reinforcement learning for detecting malicious websites. arXiv Preprint arXiv:1905.09207. https://doi.org/10.48550/arXiv.1905.09207

[11] Belfedhal, A.E., Belfedhal, M.A. (2022). A lightweight phishing detection system based on machine learning and URL features. International Conference on Managing Business Through Web Analytics, Cham: Springer International Publishing, 307-319. https://doi.org/10.1007/978-3-031-06971-0_22

[12] Le, H., Pham, Q., Sahoo, D., Hoi, S.C. (2018). URLNeT: Learning a URL representation with deep learning for malicious URL detection. arXiv Preprint arXiv: 1802.03162. https://doi.org/10.48550/arXiv.1802.03162

[13] Peng, Y.F., Tian, S.W., Yu, L.Y., Lv, Y., Wang, R.J. (2019). A joint approach to detect malicious URL based on attention mechanism. International Journal of Computational Intelligence and Applications, 18(03): 1950021. https://doi.org/10.1142/S1469026819500214

[14] Sutton, R.S., Barto, A.G. (1998). Reinforcement learning: An introduction. IEEE Transactions on Neural Networks, 9(5): 1054-1054. https://doi.org/10.1109/TNN.1998.712192

[15] Ebbu2017. Phishing Dataset. (2018). https://github.com/ebubekirbbr/pdd.

[16] Yuan, J.T., Chen, G.X., Tian, S.W., Pei, X.J. (2021). Malicious URL detection based on a parallel neural joint model. IEEE Access, 9: 9464-9472.

https://doi.org/10.1109/ACCESS.2021.3049625
[17] McGahagan, J., Bhansali, D., Pinto-Coelho, C., Cukier, M. (2019). A comprehensive evaluation of webpage content features for detecting malicious websites. In 2019 9th Latin-American Symposium on Dependable Computing (LADC), IEEE, 1-10. https://doi.org/10.1109/LADC48089.2019.8995713
[18] Wang, Y., Cai, W.D., Wei, P.C. (2016). A deep learning approach for detecting malicious javascript code. Security and Communication Networks, 9(11): 1520-1534. https://doi.org/10.1002/sec.1441
[19] Saxe, J., Harang, R., Wild, C., Sanders, H. (2018). A deep learning approach to fast, format-agnostic detection of malicious web content. In 2018 IEEE Security and Privacy Workshops (SPW), IEEE, 8-14. https://doi.org/10.1109/SPW.2018.00010
[20] Alex, S., Rajkumar, T.D. (2021). Spider bird swarm algorithm with deep belief network for malicious javascript detection. Computers & Security, 107: 102301. https://doi.org/10.1016/j.cose.2021.102301
[21] Yang, P., Zhao, G.Z., Zeng, P. (2019). Phishing website detection based on multidimensional features driven by deep learning. IEEE Access, 7: 15196-15209. https://doi.org/10.1109/ACCESS.2019.2892066
[22] Li, Y.K., Yang, Z.G., Chen, X., Yuan, H.P., Liu, W.Y. (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
[23] Ke, G., Meng, Q., Finley, T., Wang, T.F., Chen, W., Ma, W.D., Ye, Q.W., Liu, T.Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30. https://doi.org/10.5555/3294996.3295074
[24] Jabeen, S., Li, X., Amin, M.S., Bourahla, O., Li, S.Y., Jabbar, A. (2023). A review on methods and applications in multimodal deep learning. ACM Transactions on Multimedia Computing, Communications and Applications, 19(2s): 1-41. https://doi.org/10.1145/3545572
[25] Bilgera, C., Yamamoto, A., Sawano, M., Matsukura, H., Ishida, H. (2018). Application of convolutional long short-term memory neural networks to signals collected from a sensor network for autonomous gas source localization in outdoor environments. Sensors, 18(12): 4484. https://doi.org/10.3390/s18124484
[26] BERT Preprocessing with TF Text. Tensorflow.org. https://www.tensorflow.org/text/guide/bert_preprocessing_guide, accessed on Nov. 06, 2022.
[27] Lan, Z.Z., Chen, M.D., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. arXiv Preprint arXiv: 1909.11942. https://doi.org/10.48550/arXiv.1909.11942
[28] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2018). Bert: pre-training of deep bidirectional transformers for language understanding. arXiv Preprint arXiv: 1810.04805. https://doi.org/10.48550/arXiv.1810.04805
[29] Hidayat, A. (2015). Esprima. https://github.com/Kronuz/esprima-python.
[30] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv. 2016, arXiv:1607.01759. https://doi.org/10.48550/arXiv.1607.01759
[31] Singh, A.K. Goyal, N. (2017). Malcrawler: A crawler for seeking and crawling malicious websites. International Conference on Distributed Computing and Internet Technology, 10109: 210-223. https://doi.org/10.1007/978-3-319-50472-8_17
[32] Aljabri, M., Alhaidari, F., Mohammad, R.M., Mirza, S., Alhamed, D.H., Altamimi, H.S., Chrouf, S.M., Ijaz, M.F. (2022). An assessment of lexical, network, and content-based features for detecting malicious URLs using machine learning and deep learning models. Computational Intelligence and Neuroscience, 2022: 2022. https://doi.org/10.1155/2022/3241216