Enhancing Web Application Security: A Deep Learning and NLP-based Approach for Accurate Attack Detection

Pham Van Hau, Do Thi Thu Hien

***Abstract*— Nowadays, web attacks have become more complicated, leading to the difficulty of traditional web application firewalls (WAFs) in recognizing those threats, especially when dealing with new attacks. Hence, machine learning/deep learning (ML/DL) approaches have been applied to the field of web attack detection with proven success. However, most existing ML/DL-based web attack detectors focus on a specific type of attack due to the difference in the payload of various attacks, which sets a border to the capability of those solutions in detecting new attack types. In this paper, we propose a novel DL-based solution for web attack detection, named DL-WAD, leveraging deep learning and natural language processing techniques. Moreover, DL-WAD is designed with a data preprocessing mechanism aimed at differentiating between regular web requests and malicious ones that carry attack payloads encompassing multiple types of web attacks. The experiment results indicate the effectiveness of our solution in protecting the target web services from a wide range of attacks with high accuracy.**

***Tóm tắt*— Ngày nay, tấn công web ngày càng trở nên phức tạp và tinh vi, gây khó khăn cho các tường lửa ứng dụng web (WAF) truyền thống trong việc nhận diện các mối đe dọa này, đặc biệt là các kiểu tấn công mới. Do vậy, các hướng tiếp cận dựa trên học máy/học sâu (ML/DL) đã được áp dụng trong lĩnh vực nhận diện tấn công web và đạt được những thành công nhất định. Tuy vậy, do sự khác biệt trong payload của các loại tấn công khác nhau, hầu hết các bộ phát hiện tấn công web sử dụng ML/DL thường tập trung vào một loại tấn công web nhất định, đặt ra một rào cản cho các giải pháp này khi triển khai để áp dụng để phát hiện các loại tấn công mới. Trong bài báo này, nhóm tác giả đề xuất một giải pháp dựa trên DL để phát hiện tấn công web, có tên DL-WAD, sử dụng học sâu và các kỹ thuật xử lý ngôn ngữ tự nhiên. Thêm vào đó, DL-WAD được thiết kế với cơ chế tiền xử lý dữ liệu cho phép phân biệt các yêu cầu web bình thường và yêu cầu bất thường có chứa payload của nhiều loại tấn công web khác nhau. Kết quả thực nghiệm cho thấy hiệu quả của giải pháp đề xuất trong việc bảo vệ các dịch vụ web khỏi nhiều loại tấn công với độ chính xác cao.**

Keywords*— Web attack detection; deep learning; natural language processing; web application security.

Từ khóa*— Phát hiện tấn công web; học sâu; xử lý ngôn ngữ tự nhiên; bảo mật ứng dụng web.

I. INTRODUCTION

Along with the rapid development of information technology, there is a dramatic increase in the number of web-based applications in various aspects of life, ranging from news websites, and online training platforms to e-commerce. Due to their popularity and the valuable user data that they manage, web-based applications have become an attractive target for attackers to steal that information for negative purposes. A diversity of attack types, such as SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Local or Remote File Inclusion (LFI/RFI), etc. are common threats to the security of web applications that any developers or administrator have to take into account. Hence, there is a need for robust protection for web services against those threats to ensure their proper operations.

In the very first stage of this journey, Web application firewalls (WAFs) are introduced as rule-based filters customized with the knowledge of common attacks for protecting

This manuscript is received on November 21, 2023. It is commented on December 05, 2023 and is accepted on December 09, 2023 by the first reviewer. It is commented on December 06, 2023 and is accepted on December 18, 2023 by the second reviewer.

web applications [1-3]. Despite the ability to figure out malicious behaviors on web-based packets, this popular security solution exposes its weakness in dealing with increasingly sophisticated or zero-day attacks. Meanwhile, AI techniques in terms of machine learning (ML) and deep learning (DL) have achieved promising results when being applied in the fields of computer vision, which raises its potential in solving other problems, such as cybersecurity. Many efforts have been made to adapt ML/DL to detect attacks, especially for web-based applications [4-6]. The detection tasks are mainly performed on URLs in incoming HTTP requests to seek abnormal behaviors. Moreover, as URLs and their parameters or weblogs are both text-based information, those solutions need to consider the complicated relations of words that make up a valid attack payload. In this context, Natural Language Processing (NLP) techniques can offer effective ways to obtain valuable information from analyzing bunches of words. Noted this, combining DL and NLP techniques has become the idea of various works [7-9], where NLP is enabled in either data processing or models but not both of them.

Motivated by the above promising solutions, this paper aims to propose DL-WAD, a DL-based approach for detecting web attacks that takes advantage of NLP techniques in data preprocessing and the Bi-LSTM model, which is a well-known deep learning algorithm in the field of NLP.

The main contributions of this paper are as follows:

- The extended keyword list and expressions allow our proposed web attack detector to recognize the malicious payloads in URLs of more diverse attack types.

- Natural Language Processing (NLP) techniques and the Bi-LSTM model are leveraged to enhance the accuracy of our model in analyzing the language-related nature of URLs.

- We conduct experiments using 3 popular datasets in web attack detection, in which the training and testing phases are both performed on all chosen datasets instead of a single one. Moreover, the performance of our proposed Bi-LSTM model is also compared to other deep learning models as well as other related approaches.

The remainder of this work is organized as follows. In Section II, we give the background of web application firewalls and the related works of applying ML/DL in detecting web attacks. Section III presents our DL-based web attack detection system DL-WAD using DL algorithms combined with NLP techniques, which can obtain knowledge of different web attack types. The experimental settings and evaluation results of our proposed system are given in Section IV. Finally, in Section V, we conclude the paper and discuss future directions for this work.

II. Related work

A well-known solution to protect web applications from attacks is Web Application Firewalls (WAF). This kind of security software is located in front of web servers and works as a customized filter created based on knowledge about known attacks to verify every HTTP/HTTPS request and response for abnormal behaviors. Though WAFs play an essential role in ensuring the security of web applications and their data, traditional WAFs require experts in configuration to properly enable protection capability.

As the drawback of detecting web attacks based on their signature, the potential of applying AI techniques (ML and DL) in detecting web-based threats has become the motivation of multiple promising solutions.

Solomon et al. [4] focused only on SQL injection attacks in web applications and proposed an alternative ML predictive analytics to overcome the existing signature-based solution to detect this attack. A classifier using Two class Support Vector Machine was deployed to predict binary labels. Experiments performed on their created and labeled dataset proved the effectiveness of proposed method with accuracy of 98.6%.

Later, a work called DeepWAF by Kuang et al. [5] explored the ability of DL in detecting web attacks by implementing and evaluating various DL models including CNN, LSTM, and their combinations. The high detection rate of approximately 95% and low false alarm in experiments based on the CSIC2010 dataset indicated the potential of DL in ensuring web application security.

On the other hand, a research group of Shahid proposed a solution for web attack detection using enhanced DL [6]. For more details, the CNN model was trained to detect web attacks via HTTP request parameters like Type, Content length, Data, URLs, etc., and was nested with a Cookie Analysis Engine verifying all incoming cookies. This publication also suggested a request analyzer for attacker profiling to prevent the DL classifier from being triggered for every request in a real-time environment. The authors validated their proposed approach on both their custom dataset and public benchmark one – CSIC2010, and achieved an accuracy of more than 98.7%.

With the idea of applying NLP to the field of web attack detection, the work of Ming Zhang [7] designed a special architecture of CNN with an embedding layer to perform detection on URLs in HTTP requests. Their experiments proved the effectiveness of their system using CSIC2010 dataset, with 96.49% in accuracy and a low false alarm rate.

Zhihong Tain et al. [8] tried to address the issue of data centralization on edge devices by using multiple concurrent DL models and introducing a word embedding-based method to represent all kinds of URLs. Thanks to this design, the authors could train the model to detect multiple attack types via 3 public datasets including CSIC 2010, FWAF and HttpParams. They achieved 99.41% in accuracy, higher than that of other solutions mentioned in the work.

Meanwhile, the work of Hacer Karacan et al. [9] proposed Bi-LSTM-based web application security modes to detect web attacks. This DL algorithm is usually used in NLP-related tasks due to its ability to process data in both directions to better understand the relationship between sequences. Moreover, the authors also introduced a data augmentation technique to separate normal and abnormal HTTP traffics for better detection rates. In addition to well-known public datasets such as CSIC2010 and ECML/PKDD, the authors also created their own new dataset. The experiments proved the effectiveness of their work via detection rates of 98.34% and 93.91% for binary and multi-class classifications, respectively.

In another work of Seyyar et al. [10], the authors also utilized an NLP technique called BERT to analyze URLs from web requests, and then a CNN model was deployed in the classification phase to detect any malicious URLs. The proposed method was proven to be effective in classifying normal and abnormal requests in terms of accuracy (96%) and time consumption.

The above works indicate the potential of DL algorithms in effectively protecting web applications against various attacks. However, most of them applied NLP-related techniques either in processing URLs or the used DL models, not in both aspects. Our work considers NLP techniques in data processing steps and uses Bi-LSTM as the underlying model of the detector.

III. PROPOSED SYSTEM

This section describes the overall architecture of our proposed DL-based web attack detector, DL-WAD, as well as its workflow in the training and detecting phases.

A. Overall architecture

The architecture of DL-WAD is depicted in Figure 1, which is composed of components for data preprocessing and a DL-based attack detector engine.

As we can see from Figure 1, input data for the proposed system is the requested URLs in HTTP requests. Then, it is processed via multiple steps of data preprocessing including normalization, word embedding, and tokenization to be ready to be used in training

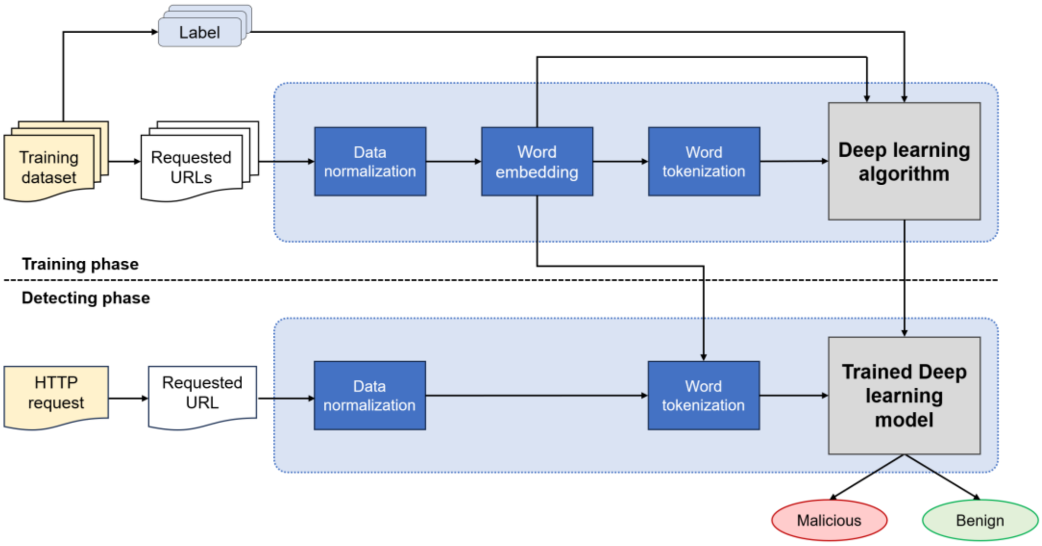


Figure 1. Overall architecture of the proposed DL-WAD

models or detecting abnormal behavior. Note that, the last 2 steps of preprocessing take advantage of NLP-related techniques to obtain more valuable information for the training and detecting phases.

B. Requested URL extracting

The main data for attack detection in our system is URLs in HTTP requests. In the case of HTTP GET requests, this data can be obtained directly from the URL part in the Request line (the first line) in that HTTP request, which also consists of its parameters in the form of `?param1=value1&param2=value2`. Meanwhile, URLs of POST requests can be reconstructed by concatenating the URL in the Request line and data in the body converted to the form of parameters as in the GET request. Note that, although there are POST requests with multiple-part data in multiple lines in the body, we only focus on those requests with single-part data in only one line. Moreover, the domain of the URL is omitted in our approach.

C. Data Preprocessing

After extracting the requested URL from an HTTP request, this information is preprocessed via 3 different sub steps to transform it into an appropriate input format for deep learning models.

1. Data Normalization

In fact, various web attacks are performed via specific forms of URLs with various annotations, resulting in differences in the structure or parameters. Moreover, attackers also try to make multiple variations of those payloads to enhance their evasion ability against detectors. Hence, this step aims to have the same representation for those URLs regardless of their underlying attacks, which is later the input for training models.

1.1. Special keyword definition

Firstly, we define a list of special keywords assigned to web development languages (HTML, Javascript, etc.) or 6 specific following web attacks focused in the scope of this paper.

(i) *SQL Injection:* As this attack aims to exploit vulnerabilities in the input sanitization process for SQL queries, some common SQL-related keywords like `union`, `and`, `or`, `order by`, and special characters such as `#`, `--`, `/*`, `'`, " can be its indicators.

(ii) *XSS:* Attackers perform this attack to send malicious JavaScript code to users. Therefore, we can recognize this attack using keywords relating to Javascript such as `alert`, `<>` or its triggers like `onerror`, `onmouseover`, `onload`.

(iii) *OS Command Injection:* In this attack, attackers can exploit the vulnerability to send and execute arbitrary OS commands on the server. Hence, the occurrence of OS commands like `pwd, ls, where, cd, ssh, ||, &&, echo, whoami` can be indicators of this attack.

(iv) *Remote File Inclusion (RFI):* This attack can take place in web applications with improperly sanitized input functions so that attackers can upload malicious files or scripts from external sources and execute them. Definitely, a reference to resources in other domains can be an indicator of this attack.

(v) *Local File Inclusion (LFI) and Path Traversal:* Attackers take advantage of "`../`" to access crucial files or folders on a web server. While the former attack also allows reading and executing files, the latter one only provides read-only access. Hence, "`../`" and its variations as well as some special file or folder names such as `etc, htpasswd, passwd` can be used to detect these attacks.

(vi) *CRLF:* Because this attack is based on the newline character combination in HTML and HTTP headers to inject malicious payloads, commonly used CRLF characters including `|0a 0d|, %0d %0a` etc. are considered indicators.

Based on the indicators of the above attacks, special keywords in our work are summarized in Table 1. These keywords are most used in attacks and should be selected by experienced experts or attack analysts. This list is then used to figure out which words are important and should remain unchanged in a URL to ensure the key characteristics of their corresponding attacks.

TABLE 1. KEYWORDS FOR DATA NORMALIZATION

Description	**Keywords**
SQL Injection	select, waitfor, delete, insert, limit, replace, drop, order by, create, desc, group by, alter,...
Javascript & XSS	alert, br, document, script, switch, await, function,...
HTML	doctype, base, area, button, audio, abbr, div, onload,...
OS commands	cp, ls, rm, cat, sh, pwd, whoami, mv, tail, more,...
Punctuations	-- / + ? & ; = , ( ) <>* ! $ # | ˆ { } \ ˜ @ . ‘ [ ] : ’ ”
File extension	.php, .html, .txt, .js, .zip, .css, .asp,...

TABLE 2. EXPRESSION IN URL TRANSFORMATION

Expression	**Description**
NullChar	Replace %00, which is the URL encoded format of Null character
LFChar	Replace %0A, which is the URL encoded format of Line Feed character
CRChar	Replace %0D, which is the URL encoded format of Carriage Return character
SpaceChar	Replace %20, which is the URL encoded format of Space character
InvisChar	Replace all invisible characters in the range of %00 - %20, except above ones
PathString	Replace the path in URLs
Numbers	Replace all numbers in URLs
PureString	Replace strings including only a-z and '-'
UniString	Replace all strings including Unicode characters
HexString	Replace all strings including hexadecimal data
MixString	Replace strings including other characters except for a-z and '-'
SenString	Replace keywords that are not in training data but in real URLs

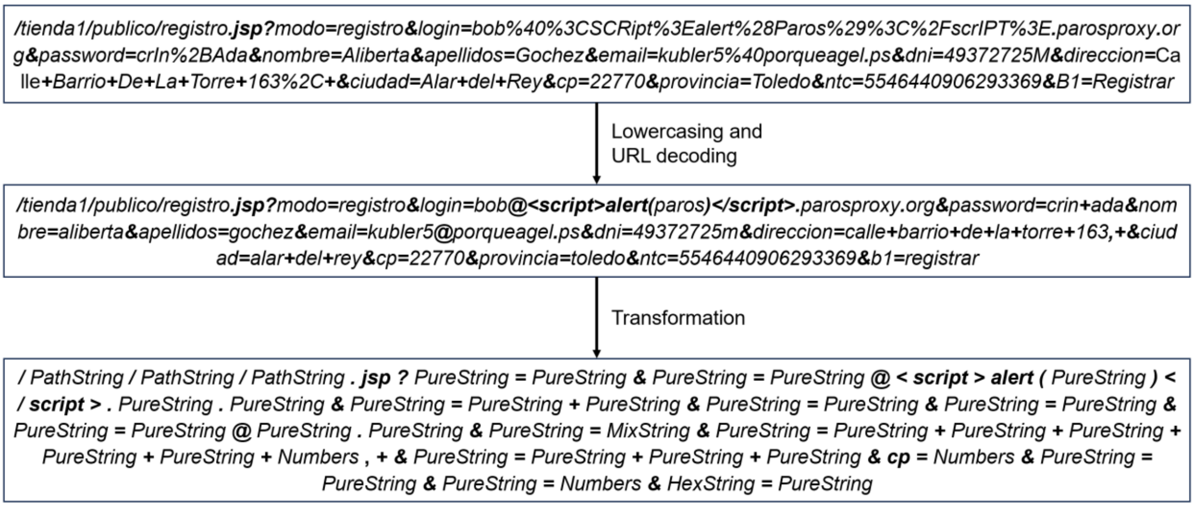
1.2. URL transformation

After lowercasing and performing URL decoding on the URL, an n-1 transformation scheme is then applied to decoded URLs to convert words into expressions. Note that, predefined keywords and punctuations are unchanged in this step due to their vital roles in the function of attacks. Meanwhile, the remaining words with the same properties are transformed into the same expressions as in Table 2.

An example processing result of these 2 sub-steps is given in Figure 2. As we can see, along with numerous punctuation remained, there are 5 keywords including `.jsp`, `<script>`, `alert()`, `cp` and `</script>` left untouched in the URLs.

2. Word embedding

Word embedding is a word representation technique to ensure that similar words are presented in the same ways. Among various



```
/tienda1/publico/registro.jsp?modo=registro&login=bob%40%3CSCRipt%3Ealert%28Paros%29%3C%2FscrIPT%3E.parosproxy.org&password=crIn%2BAda&nombre=Aliberta&apellidos=Gochez&email=kubler5%40porqueagel.ps&dni=49372725M&direccion=Calle+Barrio+De+La+Torre+163%2C+&ciudad=Alar+del+Rey&cp=22770&provincia=Toledo&ntc=5546440906293369&B1=Registrar
```

```
/tienda1/publico/registro.jsp?modo=registro&login=bob@<script>alert(paros)</script>.parosproxy.org&password=crin+ada&nombre=aliberta&apellidos=gochez&email=kubler5@porqueagel.ps&dni=49372725m&direccion=calle+barrio+de+la+torre+163,+&ciudad=alar+del+rey&cp=22770&provincia=toledo&ntc=5546440906293369&b1=registrar
```

```
/ PathString / PathString / PathString . jsp ? PureString = PureString & PureString = PureString @ < script > alert ( PureString ) < / script > . PureString . PureString & PureString = PureString + PureString & PureString = PureString & PureString = PureString & PureString = PureString @ PureString . PureString & PureString = MixString & PureString = PureString + PureString + PureString + PureString + PureString + Numbers , + & PureString = PureString + PureString + PureString & cp = Numbers & PureString = PureString & PureString = Numbers & HexString = PureString
```

Figure 2. An example of data normalization in URL preprocessing

methods for word embedding, Word2vec [10] models perform the mapping of words to vectors as the representation. In the scope of this paper, the Word2vec model called CBOW [10] - a neural network for NLP tasks, takes the duty of this step.

CBOW learns to predict the target word based on the context of surrounding ones, tweaking the weight matrixes. Taking the normalized URLs as the input, the CBOW model generates numerical vectors that capture the semantics of words in a continuous vector space. Note that, transformed input URLs in this step have been normalized to contain only keywords and similar expressions, not the actual words. As a result, the matrix only contains weights for those predefined keywords and expressions (Table 2). Hence, the size of this matrix depends on the number of keywords and expressions.

Besides, word embedding is only performed in the training phase of our model to output the weight matrix based on the given training dataset. Once the training is done, a weight matrix is obtained to be used in later steps as well as the detecting phase.

3. Word Tokenization

This step aims to transfer requested URLs from vectors of words into numerical form for compatibility with deep learning models. To achieve this, the resulting weight matrix from the CBOW model is used to map words to numbers. For more details, each word is assigned to its corresponding index in this matrix, and then its appearance in all URLs. In consequence, the result of this step is representative numerical vectors of URLs, in which the same keywords and expressions in URLs are replaced with the same numbers.

Moreover, the lengths of URLs are often different from each other due to various structures or parameters, and so are their vectors, which is unacceptable by deep learning models. Hence, those vectors are padded with values of 0 to achieve the same length, which is the length of the longest URL in the dataset.

D. DL-based attack detector

This component in our DL-WAD is designed as a deep learning model trained to have the capability of recognizing benign URLs and malicious ones. This paper chooses Bidirectional Long Short-Term Memory (Bi-LSTM) [12], a deep learning model relating to NLP, which has achieved potential results in classification. The model architecture of Bi-LSTM used in this paper is described in Figure 3.

Among various deep learning architectures, the Bi-LSTM model emerges due to its unique capabilities and advantages compared to other models like RNN, LSTM, GRU, CNN, and DNN. Unlike traditional RNN, Bi-LSTM effectively addresses the vanishing gradient issue, enabling the capture of longer dependencies within sequences. Compared to LSTM and GRU, Bi-LSTM outpaces by processing sequences bidirectionally, allowing

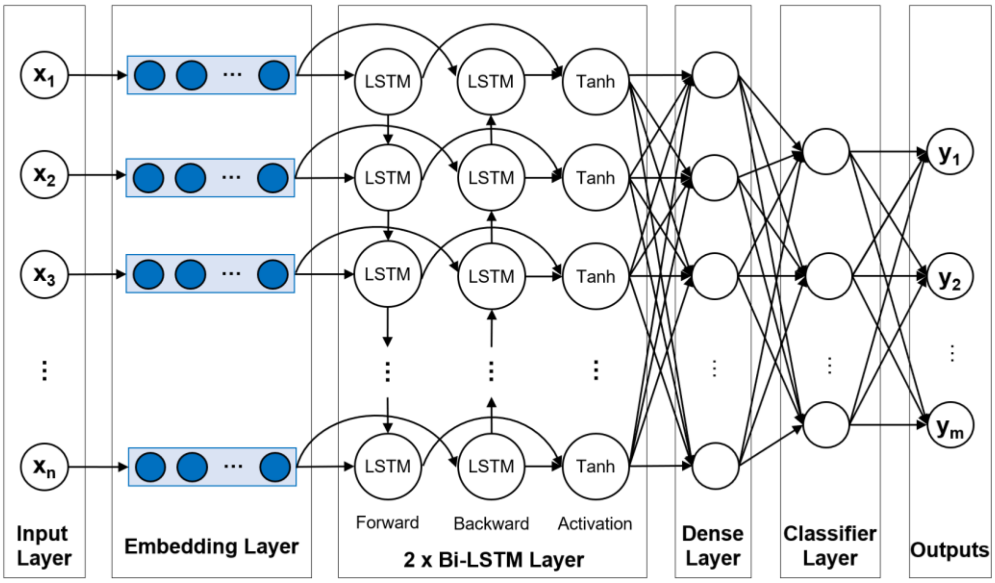


Figure 3. The Bi-LSTM model of DL-WAD

them to harness context from both past and future elements. This attribute gives them a notable edge in feature extraction and pattern recognition. While CNN stands out for spatial feature extraction, Bi-LSTM specializes in tasks involving sequential data, such as natural language processing, speech recognition, and time series analysis. Unlike the DNN, which lacks the ability to model sequence dependencies, Bi-LSTM offers a comprehensive solution to sequence analysis, making them particularly effective in tasks where understanding intricate temporal relationships is essential.

The utilization of Bi-LSTM models presents a significant advancement in the realm of deep learning, particularly for tasks that involve the analysis of sequential data. One of the standout advantages of Bi-LSTM lies in its ability to capture contextual information from both past and future elements within a sequence. This dual-direction processing enables a more comprehensive understanding of intricate relationships and dependencies inherent in the data. Moreover, Bi-LSTM excels in handling long-term dependencies, effectively mitigating issues related to vanishing gradients and substantially enhancing their capacity to recognize intricate patterns over extended sequences. As a result, Bi-LSTM proves to be particularly valuable in applications where accurate labeling or prediction hinges on the nuanced context surrounding each data point. Furthermore, their adaptability to varying sequence lengths, coupled with their inherent regularization capabilities, renders Bi-LSTMs a robust choice for the task of this research. The transformed URLs have forms of sentences that are similar to those of NLP.

In the proposed model, a transformed URL is put into the Embedding layer for vector conversion as input of the two Bi-LSTM layers. Each Bi-LSTM layer consists of a forward LSTM layer, a backward LSTM layer, and an activation layer of the Tanh function. Besides, the activation layer fully connects to a Dense layer, followed by a classifier layer. Finally, outputs are values to determine the type of HTTP request.

IV. EXPERIMENTS

This section presents experimental settings and datasets used to evaluate the effectiveness of the proposed DL-WAD.

A. *Experimental setup*

1. *Implementation*

The proposed DL-WAD is implemented on an Ubuntu 22.04 virtual machine with the hardware configuration of Intel Core i7-7700 CPU, 16GB RAM, and Python as well as its supporting ML/DL libraries and frameworks such as TensorFlow, gensim, scikit-learn installed.

2. The structure of proposed model

The proposed Bi-LSTM model includes an embedding layer, two continuous Bi-LSTM layers, one fully connected layer, and one softmax layer for binary classification, illustrated in Table 3**.** Moreover, our model is trained in 100 epochs with a batch size of 256 and a learning rate of 0.001. In terms of NLP techniques, the generated vectors representing analyzed URLs have the size of 50.

TABLE 3. DETAILS OF PROPOSED BI-LSTM MODEL STRUCTURE

Layer (Type)	**Input shape**	**Output shape**	**Activation function**
Input	(m, n)	(m, n)	-
Embedding	(m, n)	(m, n, 50)	-
Bi-LSTM	(m, n, 50)	(m, n, 256)	tanh
Bi-LSTM	(m, n, 256)	(m, 512)	tanh
Dense	(m, 512)	(m, 256)	relu
Dense	(m, 256)	(m, 2)	softmax

3. Dataset

To evaluate the ability to detect multiple web attacks of DL-WAD, 3 different public datasets, called HTTP Dataset CSIC 2010, FWAF, and HttpParams with various attack types such as Command injection, CRLF, File Disclosure, SQL injection, XSS, LFI, RFI, etc., are used in experiments.

- HTTP Dataset CSIC 2010 [13], which was created by the CSIC research group of Carlos III University, consists of more than raw 36,000 HTTP requests obtained from 150 e-commerce and news websites, including various attack types such as SQL Injection, XSS, CRLF Injection and File disclosure.

- FWAF dataset [14] was created by the FSecurity community to evaluate the effectiveness of ML-based web firewalls. This dataset is made of extracted URLs from over 1,290,000 normal requests and 48,000 malicious requests from the Internet with common web attacks such as XSS or SQL Injection.

- HttpParams [15] is a popular dataset on Github, generated by using various tools such as SQLMap, XSSYA, Vega Scanner, and the FuzzDB repository. Note that, instead of containing full HTTP requests, this dataset only includes the payloads from 19,304 normal requests and 11,763 abnormal requests which are labeled as 'norm' and 'anom' correspondingly. Moreover, requests of various attack types including SQL Injection, XSS or Command Injection can be found in this dataset.

Besides, while those datasets are different in record form, our proposed DL-WAD is designed to work with URLs in requests. Hence, except CSIC 2010 which already has its data in the appropriate format, we perform additional tasks on FWAF and HttpParams datasets to extract URLs. For more details, while FWAF with raw HTTP requests is preprocessed with the method mentioned in Section III.B, the payloads in HttpParams are combined with a randomly generated path to form a URL. Moreover, those datasets are filtered by removing duplicated or incorrectly labeled records to eliminate their negative impact on training performance.

The distribution of records in 3 datasets after this task is given in Table 4. Later, those datasets are divided into 2 different parts for training and testing, with a ratio of 80:20.

TABLE 4. DATA DISTRIBUTION IN DATASETS

Dataset	**No. of normal records**	**No. of abnormal records**
CSIC2010	4,704	5,686
FWAF	48,976	44,458
HttpParams	19,135	11,759
Total	***72,815***	***61,903***

B. Evaluation metrics and scenarios

1. Metrics

To evaluate the effectiveness of our proposed DL-WAD in detecting web attacks, we observe the various metrics of Accuracy, Precision, Recall, and F1-Score. Given TP, TN, FP, and FN stand for True Positive, True Negative, False Positive, and False Negative, respectively, the 4 mentioned metrics are defined as follows:

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

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

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

$$F1\text{-}score = \frac{2\times Precision \times Recall}{Precision + Recall} \qquad (4)$$

2. Scenarios

Two scenarios are designed as follows:

- *Scenario 1:* This scenario aims to verify the capability of detecting web attacks of our proposed model, which uses Bi-LSTM after being trained with the 3 mentioned datasets. In detail, those datasets are combined to form a total one, of which 80% is used as the training set for the Bi-LSTM model. Then, in the testing phase, we have 3 different cases where the remaining 20% of each dataset is utilized as the testing set to evaluate the trained model.

- *Scenario 2:* In this scenario, in addition to the chosen Bi-LSTM, other 6 ML and DL models, including CNN, RNN, DNN, LSTM, GRU and M-ResNet from [8], are implemented for comparison to figure out the best one in attack detecting performance. The training scheme in this scenario is still the same as the previous one with 80% of the combined dataset, while in the testing step, we use only the remaining CSIC-2010 dataset in this scenario.

TABLE 5. EXPERIMENTAL RESULTS OF THE PROPOSED MODEL ON THE TESTING SET USING THREE DATASETS

Testing set	Accuracy	Precision	Recall	F1-score
CSIC2010	0.9745	0.9909	0.9621	0.9763
FWAF	0.9956	0.9981	0.9927	0.9954
HttpParams	**0.9991**	**0.9991**	**0.9987**	**0.9989**

C. Evaluation results

The experimental scenarios on the three datasets are evaluated with the metrics mentioned in the previous section.

1. Scenario 1

The observed metrics reflecting the performance of Bi-LSTM-based DL-WAD in detecting attacks are summarized in Table 5. Overall, our model achieves excellent performance with metrics over 0.96 regardless of the used testing datasets. Moreover, the detecting capability of our proposed model on the HttpParams dataset outperforms other counterparts with the best figures in all evaluating metrics. Despite the slightly worse performance compared to the testing case with HttpParams, performance on FWAF also witnesses impressive figures which are higher than 0.99. Clearly, though those datasets consist of patterns of multiple attack types, our model still achieves good knowledge to be able to distinguish them in the testing sets with high accuracy.

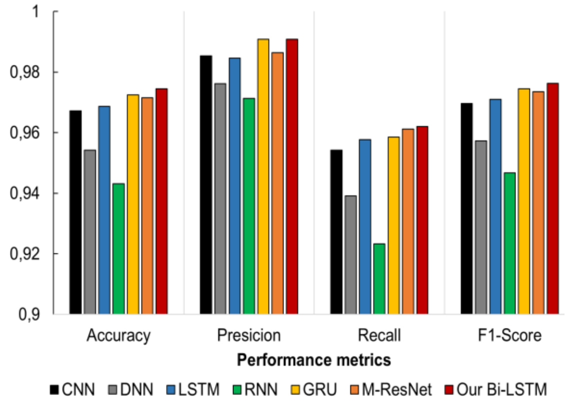


Figure 3. Comparison in attack detection performance of Bi-LSTM-based model and various DL model.

2. Scenario 2

Table 6 gives the result when putting our proposed model side by side with other DL models, while the virtualization of the same result can be found in Figure 4.

Obviously, using the same given CSIC-2010 dataset, the Bi-LSTM model in our proposed DL-WAD takes the lead in all metrics, which is also depicted in Figure 4 with the highest red columns. The second most effective model is GRU, while RNN experiences the worst performance in all 4 metrics.

TABLE 6. COMPARISON RESULTS OF THE PROPOSED DL-WAD MODEL WITH OTHER DL MODELS AND WORKS ON THE CSIC-2010 DATASET

Model	Accuracy	Precision	Recall	F1-score
CNN	0.9672	0.9854	0.9542	0.9696
DNN	0.9542	0.9762	0.9392	0.9573
LSTM	0.9687	0.9846	0.9577	0.9710
RNN	0.9432	0.9713	0.9233	0.9467
GRU	0.9725	**0.9909**	0.9586	0.9745
M-ResNet [7]	0.9716	0.9864	0.9612	0.9736
DL-WAD	**0.9745**	**0.9909**	**0.9621**	**0.9763**

V. CONCLUSION

In this study, we propose a web attack detection architecture leveraging the Bi-LSTM model and natural language processing. This architecture enables the protection of web applications from different types of attacks. In addition, the data preprocessing and normalization transform URLs and payloads into a suitable form to be used as inputs for the model. The experimental results prove the obvious effectiveness of the combination of word2vec and DL models. Moreover, our architecture has the potential ability for web attack detection systems in real cases due to the proposed feature extraction and transformation.

In the future, it is essential to improve the detection rate and reduce the time consumption for training and testing processes. Besides, multiple-class detection is another consideration to enhance the protection capability of the system.

REFERENCES

[1] R. A. Muzaki, O. C. Briliyant, M. A. Hasditama and H. Ritchi, "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall," in 2020 International Workshop on BigData and Information Security, 2020.

[2] S. Prandl, M. Lazarescu and D.-S. Pham, "A Study of Web Application Firewall Solutions," in International Conference on Information Systems Security (ICISS 2015), 2015.

[3] Dau, H. X., Trang, N. T. T., & Hung, N. T. (2022). A Survey of Tools and Techniques for Web Attack Detection. Journal of Science and Technology on Information Security, 1(15), 109-118. https://doi.org/10.54654/isj.v1i15.852.

[4] S. O. Uwagbole, W. J. Buchanan and L. Fan, "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention," in 2017 IFIP/IEEE (IM), 2017.

[5] X. Kuang, M. Zhang, H. Li, G. Zhao, H. Cao, Z. Wu and X. Wang, "DeepWAF: Detecting Web Attacks Based on CNN and LSTM Models," in International Symposium on Cyberspace Safety and Security (CSS 2019), 2019.

[6] W. B. Shahid, B. Aslam, H. Abbas, S. B. Khalid and H. Afzal, "An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling," Journal of Network and Computer Applications, vol. 198, 2022.

[7] M. Zhang, B. Xu, S. Bai, S. Lu and Z. Lin, "A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN," in International Conference on Neural Information Processing (ICONIP 2017), 2017.

[8] Z. Tian, C. Luo, J. Qiu, X. Du and M. Guizani, "A Distributed Deep Learning System for Web Attack Detection on Edge Devices," IEEE Transactions on Industrial Informatics, vol. 16, 2020.

[9] H. Karacan and M. Sevri, "A Novel Data Argumentation Technique and Deep Learning model for Web Application Security," IEEE Access, vol. 9, 2021.

[10] Y. E. Seyyar, A. G. Yavuz and H. M. Ünver, "Detection of Web Attacks Using the BERT Model," in 2022 30th Signal Processing and Communications Applications Conference (SIU), 2022.

[11] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv, 2013.

[12] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," IEEE Transactions on Signal Processing, vol. 45, 1997.

[13] C. T. Giménez, A. P. Villegas and G. Á. Marañón, "HTTP DATASET CSIC 2010," Available: https://www.isi.csic.es/dataset/. [Accessed 2023].

[14] "Fwaf Machine Learning driven Web Application Firewall," 2017. [Online]. Available: https://github.com/faizann24/Fwaf-Machine-Learning-driven-Web-Application-Firewall . [Accessed 2023].

[15] "HttpParams Dataset," [Online]. Available: https://github.com/Morzeux/HttpParamsDataset. [Accessed 2023].

ABOUT THE AUTHORS

Pham Van Hau

Workplace: Information Security Lab, University of Information Technology, Vietnam National University Ho Chi Minh City.
Email: haupv@uit.edu.vn
Education: Received his Bachelor degree in Computer Science from the Ho Chi Minh City University of Science in 1998 and Master degree in Computer Science from the Institut de la Francophonie pour l'Informatique (IFI) in Vietnam from 2002 to 2004. He then persuaded his PhD thesis on network security under the direction of Professor Marc Dacier from 2005 to 2009.
Recent research interests: Network security; System security; Mobile security; Cloud computing.

Cơ quan làm việc: Phòng thí nghiệm An toàn thông tin, Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh.
Email: haupv@uit.edu.vn
Quá trình đào tạo: Nhận bằng Cử nhân Khoa học máy tính tại Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh vào năm 1998; Thạc sĩ Khoa học máy tính tại Viện Pháp ngữ pour l'Informatique (IFI) Việt Nam vào năm 2004; Tiến sĩ An ninh mạng dưới sự hướng dẫn của Giáo sư Marc Dacier từ năm 2005 đến năm 2009.
Hướng nghiên cứu hiện nay: An ninh mạng; Bảo mật hệ thống; Bảo mật di động; Điện toán đám mây.

Do Thi Thu Hien

Workplace: Information Security Lab, University of Information Technology, Vietnam National University Ho Chi Minh City.

Email: hiendtt@uit.edu.vn

Education: Received the B.Eng degree in Information Security and MSc degree in Information Technology from the University of Information Technology, Vietnam National University Ho Chi Minh City (UIT-VNUHCM) in 2017 and 2020, respectively.

Recent research interests: Information security and privacy; Malware; Software-defined networking and its related security-focused problem.

Cơ quan làm việc: Phòng thí nghiệm An toàn thông tin, Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh.
Email: hiendtt@uit.edu.vn

Quá trình đào tạo: Nhận bằng Cử nhân Kỹ thuật chuyên ngành An toàn thông tin và Thạc sĩ Công nghệ thông tin tại Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh lần lượt vào năm 2017 và 2020.
Hướng nghiên cứu hiện nay: Bảo mật và quyền riêng tư thông tin; Mã độc; Mạng được xác định bằng phần mềm và vấn đề tập trung vào bảo mật liên quan của nó.