BL-IDS: Detecting Web Attacks Using Bi-LSTM Model Based on Deep Learning

Saiyu Hao[1], Jun Long[1(✉)], and Yingchuan Yang[2]

[1] College of Computer, National University of Defense Technology, Changsha, China
{haosaiyu17,junlong}@nudt.edu.cn
[2] Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China

Abstract. Current anomaly-based network attack detection methods face difficulties such as unsatisfied accuracy and lack of generalization. The Rule-based Web attack detection is difficult to combat against unknown attacks and is relatively easy to bypass. Therefore, we propose a new method to detect Web attacks using deep learning. The method is based on analyzing HTTP request, where only some preprocessing is required, and the automatic feature extraction is done by the Bi-LSTM itself. The experimental results on the dataset HTTP DATASET CSIC 2010 show that the Bi-LSTM has good performance. This method has achieved state-of-the-art results in detecting Web attacks, and has a high detection rate while maintaining a low false alarm rate.

Keywords: Web attacks · Deep learning · Bidirectional long-short term memory

1 Introduction

Web applications play an extremely important role in people's daily lives. It brings great convenience to people. They can use Web applications for shopping, office, learning, entertainment and so on. However, the security of Web applications has long existed. Hackers can steal user's private data by attacking Web applications, disabled Web services, steal sensitive user information, and bring serious financial loss to both service providers and users.

However, it's hard to protect Web applications from attack. Even though developers and researchers have developed many solutions, like Web application firewalls (WAF), Web intrusion detection systems (Web IDSs), penetration testing, to protect Web applications, Web attacks remain a major threat. Generally, There are two approaches to detect Web attacks, one is the signature-based [1], another is the anomaly-based [2]. The signature-based method establish the detection model from known attacks and any behavior having the same attack signatures is identified as an attack. Contrarily, the anomaly-based method

Supported by the National Natural Science Foundation of China (Grant No. 61105050).

J. Li et al. (Eds.): SPNCE 2019, LNICST 284, pp. 551–563, 2019.
https://doi.org/10.1007/978-3-030-21373-2_45

establishes a profile from normal behaviors and any violation is identified as an attack. The signature-based method is accepted and adopted more wildly than the anomaly-based method because generally the signature-based one has lower false alarm rate and achieves higher accuracy. Although it is effective, the rule-based method is still problematic. On the one hand, It is just as good as the range of the rule set, which means it is unable to identify attacks which are not in its signature dataset. On the other hand, bypassing WAF can be done easily if they replace keywords of existing malicious requests or encode themselves multiple times [3,4].

Here, based on the BRNN [5] (Bidirectional recurrent neural networks) with the Bi-directional Long-Short Term Memory (Bi-LSTM) unit, we put forward a new anomaly detection method to detect Web attacks. Our model takes Uniform Resource Locators (URLs) and request body in the HTTP POST requests (only URLs for HTTP GET requests) as the input. After the URLs are tokenized, they will be mapped to vectors. Then the Bi-LSTM will learn from the normal request patterns. And then a trained neural network based on the output of the Bi-LSTM to judge whether given requests are anomalous. Our method has achieved state-of-the-art results in detecting Web attacks, the experimental results show that BL-IDS has a high detection rate and maintains a low false alarm rate.

The rest of the paper is organized as follows: Sect. 2 is introduction of Some related works. Section 3 is description of the method based on deep learning to detect Web attacks. Section 4 is experimental results and discussions. Section 5 is conclusion of this paper.

2 Related Works

Many machine learning techniques are used to detect Web attacks, Kruegel et al. have presented a multi-model method to detect Web attacks in [6]. The method analyzes HTTP requests and uses some different models built on different features, like attribute length, attribute character distribution, structural inference, invocation order and so on. Abou-Assaleh et al. [7] explored the idea of automatically detecting new malicious code using the collected dataset of the benign and malicious code which is based on N-gram. Moh et al. [8] have put forward a multi-stage log analysis architecture, which combines both pattern matching and supervised machine learning methods. It uses logs produced by the application during attacks to detect detecting SQL injection attacks effectively. Cao built a system which can avoid false negatives and enhance the efficiency of detecting work by using a prevailing machine learning algorithm called Adaboost in [9].

In recent years, deep learning, a branch of machine learning, has become increasingly popular and has been used in the field of information security. Cui et al. [10] propose an improved NIDS using word embedding-based deep learning (WEDL-NIDS), which first reduces the dimension of a packets payload via word embedding and learns the local contentful features of network traffic using deep convolutional neural networks (CNNs) [11], followed by adding the head features and learning global temporal features using long short-term memory (LSTM) [12]

networks. The result they got was quite well. Fredrik Valeur et al. [13] had developed an anomaly-based system that learns the profiles of the normal database access performed by Web-based applications using a number of different models. These models allow for the detection of unknown attacks with reduced false positives and limited overhead. Zhang et al. [14] have put forward a deep learning method to detect Web attacks which is using a specially designed CNN. Similar to our work, the difference is the network architectures, they use the Convolutional Neural Network while we use the Bi-LSTM based on Bidirectional recurrent neural network [5]. And the method we have proposed has better performance.

3 BL-IDS

BL-IDS aims to detect Web attack from HTTP request to improve the accuracy of IDS. Bi-LSTM can be trained using all available input information in the past and future of a given period of time, word2vec can output high-quality word vectors from huge dataset and maintain the similarity of semantic words, so we combine the advantages of both. The implementation schemes are illustrated in Fig. 1.

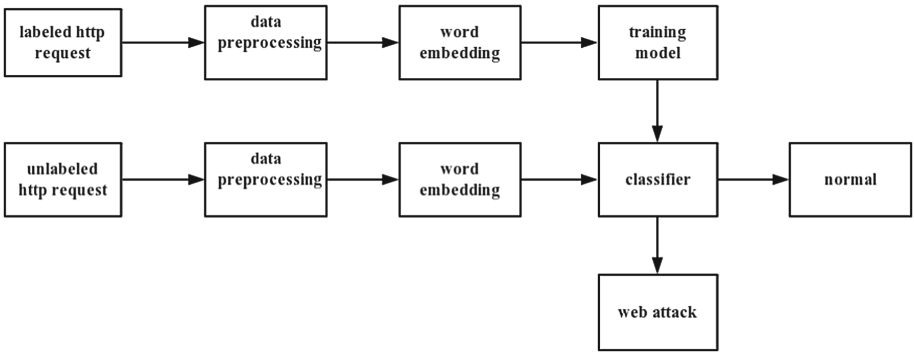


Fig. 1. The implementation schemes of Bi-LSTM

The different stages of BL-IDS are described as follows:

- **Data Preprocessing**: We decode HTTP request, then we split the decoded HTTP request, the Segmentation character includes /, & and so on.
- **Word embedding**: We map each word into a word vector using word2vec [15], The mapped word vectors are used as an input to a model based on a neural network.
- **Training model and detect Web attack**: We use the labeled word vectors to train a model based on neural network. Then use the trained model to classify the new HTTP request as Web attack or normal.

3.1 Data Preprocessing

In this section, We decode HTTP request, then we split the decoded HTTP request, the Segmentation character includes /, &, +, ?, =, @ and so on.

The process of this section is shown as Fig. 2.

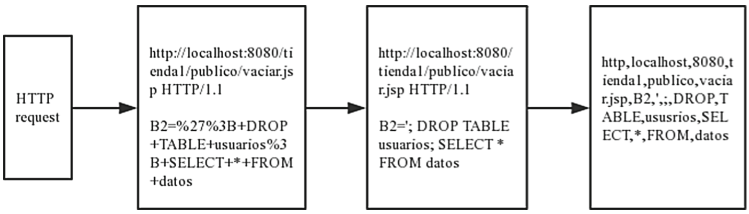


Fig. 2. The process of data preprocessing

The following is a HTTP request between a user and a Web application (Fig. 3):

```
POST http://localhost:8080/tienda1/publico/autenticar.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=23391DBBADEC19FE01E02D201F278C6A
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 60

modo=entrar&login=caria&pwd=egipciaca&remember=off&B1=Entrar
```

Fig. 3. A POST request message example by users

This is a request message based on HTTP. The HTTP request consists of three parts: the request line, headers and request body. The request line is the first line of the HTTP request message, and its format is as follows:

Method Request-URI HTTP-Version

Method represents the request method; Request-URI is a uniform resource identifier; HTTP-Version represents the requested HTTP protocol version. There are many kinds of methods. The two common methods are as GET and POST. GET request to get the resource identified by the Request-URI, POST appends new data to the resource identified by the Request-URI. The format of the Request-URI is as follows:

http://host[":"port][abs_path]

HTTP indicates that the network resource is located through the HTTP protocol; Host indicates the legal Internet host domain name or IP address; Port specifies a port number, and if it is empty, the default port 80 is used; Abs_path specifies the URI of the requested resource. HTTP/1.1 is a version of the protocol version. Headers is the additional information that the client passes the request to the server and the information of the client itself. The request body is usually the form content submitted by the user in the POST mode. The HTTP request between the hacker and the server may be like Fig. 4, and the main difference of the HTTP request between the hacker's and the user's has been bolded.

POST http://localhost:8080/tienda1/publico/vaciar.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=71D6797908C1D911A839D8BD161473AA
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 77

B2=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos

Fig. 4. A POST request message example by hackers

The main difference between the two is in the url part of the request line and the request body part (for the GET method, the main difference is in the url part), and the rest of the information we do not pay attention to. The reason for this is as follows: Most Web attacks are implemented by modifying the URL and

request body, and doing so is convenient. Taking the HTTP request message of Fig. 2 as an example, our attention is:

http://localhost:8080/tienda1/publico/vaciar.jsp
B2=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos

First, the contents of our concern is URL decoding, We can get the following:

http://localhost:8080/tienda1/publico/vaciar.jsp
B2='; DROP TABLE usuarios; SELECT * FROM datos

After that, use some special characters to divide, these special characters include:,/, &, =, +, etc.

The split data is as follows:

http, localhost, 8080, tienda1, publico, vaciar.jsp,
B2, ', ;, DROP, TABLE, ususrios, SELECT, *, FROM, datos.

3.2 Word Embedding

The effective representation of words in HTTP request is a critical step. In this section, we map each word into a word vector using Word2Vec [15]. Word embedding is a key technique in the field of natural language processing, which maps words into a vector. The mapped word vector can usually be used as an input to a neural network. Nowadays, more and more people adopt distributed representations of words in a vector space, because it can help learning algorithms to achieve better performance in natural language processing tasks by grouping similar words. Word2Vec is a excellent toolkit based on distributed representations of words and phrases. Word2Vec can output high-quality word vectors from huge data sets. At the same time, it can also maintain the similarity of semantic words, that is, the distances after similar words are mapped into vectors are similar. Therefore, in this paper, we adopt the model based on Skip-Gram. As shown in Fig. 5, the preprocessed HTTP request is mapped to a vector by Word2Vec.

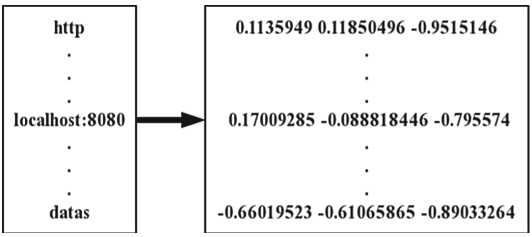


Fig. 5. The preprocessed HTTP request is mapped to a vector

Two popular implementations of Word2Vec are CBoW model and Skip-Gram model. In this paper, we adopt Skip-gram model. The difference between CBoW and Skip-Gram is that for a given context, the CBoW predicts input word,

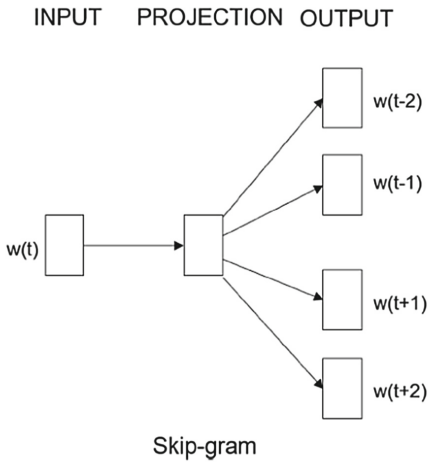


Fig. 6. The architecture of Skip-Gram model [15]

while Skip-Gram predicts the context for a given input word. The architecture of Skip-Gram model is shown in Fig. 6.

The Skip-Gram model is actually divided into two parts. The first part is to build a model, and the second part is to get embedded word vectors through the model.

3.3 Training Model and Detecting Web Attack

We treat the preprocessed sequence in Sect. 3.1 as a word, map it to a vector using word2vec as an input of model. And then we train model based on neural network use train sample. When the model is trained, it can be a classifier to detection Web attack or normal request. The neural network architecture is shown as Fig. 7.

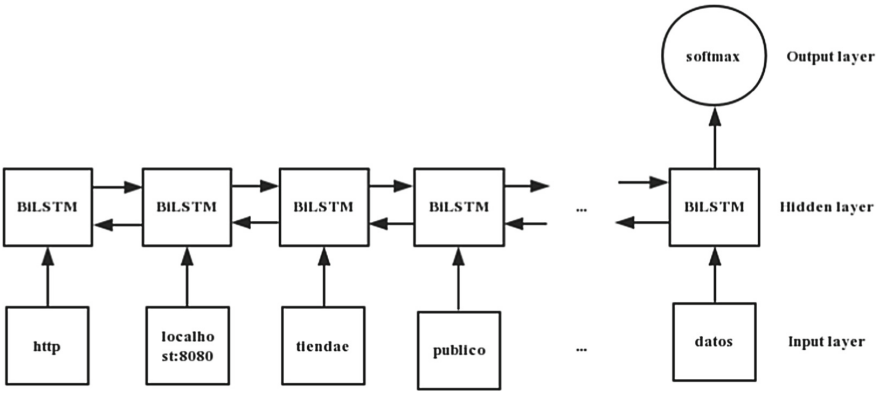


Fig. 7. The neural network architecture

The neural network model we adopted is Bi-LSTM. Bi-LSTM is based on LSTM and bidirectional recurrent neural network [5]. LSTM aims to overcome vanishing gradient problem of RNN and uses a memory cell to present the previous timestamp. The details of the memory cell in LSTM is shown in Fig. 8.

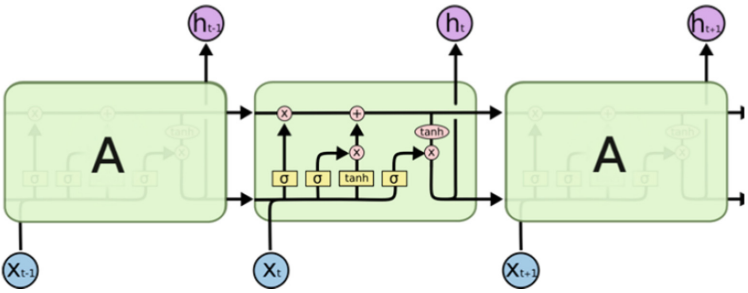


Fig. 8. The memory cell in LSTM.

Current improved LSTM usually consists of three gates in each cell: input, forget, and output. They are calculated as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{1}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \tag{2}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = f_t \cdot \tanh(C_t) \tag{6}$$

where x_t is the input at time t, W_i, W_C, W_f, W_b are weight matrices, b_i, b_C, b_f, b_o are biases, $C_t, \tilde{C}_t$ are the new state and candidate state of memory cell, f_t, o_t are forget gate and output gate.

As we all know, LSTM has achieved considerable success on many issues. But LSTM can only infer the results based on the previous information, and sometimes it is not enough to just look at the previous information. In order to detect Web attacks more efficiently and accurately, we not only need to look at the previous information, but also the information behind, so we took a Bi-LSTM which is based on bidirectional recurrent neural network. The general structure of the bidirectional recurrent neural network is shown in Fig. 9.

A bidirectional recurrent neural network (BRNN) can be trained using all available input information in the past and future of a given period of time. Therefore, it can overcome the limitations of the conventional RNN.

Finally, we use the softmax layer. The softmax classifier is used to determine whether the input is normal request or Web attack based on the vectors. For a

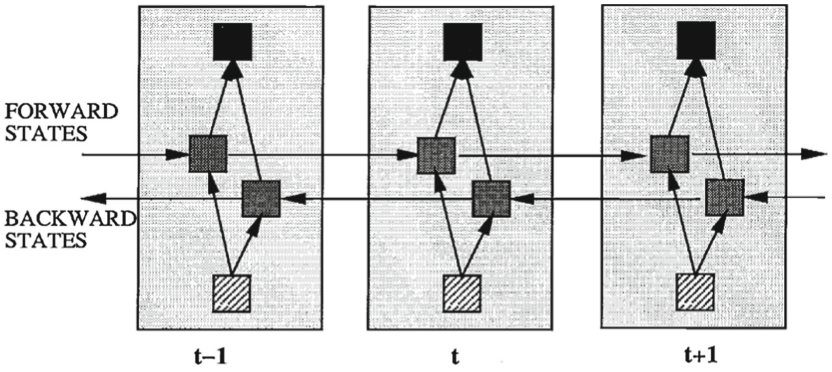


Fig. 9. General structure of the bidirectional recurrent neural network (BRNN) shown unfolded in time for three time steps [5]

list z, z_j, means the j-th element in z, and we set the output function as Softmax function:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{7}$$

where K denotes the numbers of different labels.

4 Experimental Results and Discussion

This section we conducted various experiments on the dataset HTTP DATASET CSIC 2010 [16] to evaluate the performance of our proposed method for detecting Web attacks.

4.1 Dataset

The HTTP dataset CSIC 2010 includes thousands of automatically generated Web requests which can be used to test Web attack protection systems. It was developed at the Information Security Institute of CSIC (Spanish Research National Council). The HTTP dataset CSIC 2010 includes the generated traffic targeted to an e-Commerce Web application. The dataset includes 36,000 normal requests and more than 25,000 anomalous requests. The HTTP requests are labeled as normal or anomalous.

4.2 Experiment

Experiment Setup. Our experiment software platform uses the Keras (using tensorflow as backend), all the experiments run on a server machine, whose operating system is Ubuntu 14.04. The batch size is 128 and training time is about 10 epochs. The network summary is shown in Table 1. The number of total params is 2060442, all of them is trainable.

Table 1. The network summary of BL-IDS

Layer (type)	Output shape	Param
Embedding	(None,56,40)	2053840
Bidirection	(None,56,20)	4080
Dropout	(None,56,20)	0
Bidirection	(None,20)	2480
Dropout	(None,20)	0
Dense	(None,2)	42

Evaluation Metrics. There are five metrics used to evaluate the performance of BL-IDS detecting Web attacks method: the detection rate, the false alarm rate, the accuracy, the precision and F1 score. According to the commonly used concepts in machine learning methods, we use TP, FP, TN and FN to express the number of true positive, false positive, true negative and false negative respectively. The binary confusion matrix is shown in Table 2. Detection rate (also known as recall rate) and Precision are used to evaluate the system's performance in detecting abnormal HTTP requests. False alarm rate is used to evaluate the misclassifications of normal HTTP requests. Accuracy is used to evaluate the overall performance of the system. The F1 score is used to evaluate the performance of every class of HTTP request, taking into account both precision and detection rate of the classification model. The five criteria formulas are presented below.

Table 2. Binary confusion matrix

	Actual class:abnormal	Actual class:normal
Predicted class:abnormal	TP	FP
Predicted class:normal	FN	TN

$$Detcction\ rate = \frac{TP}{TP + FN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

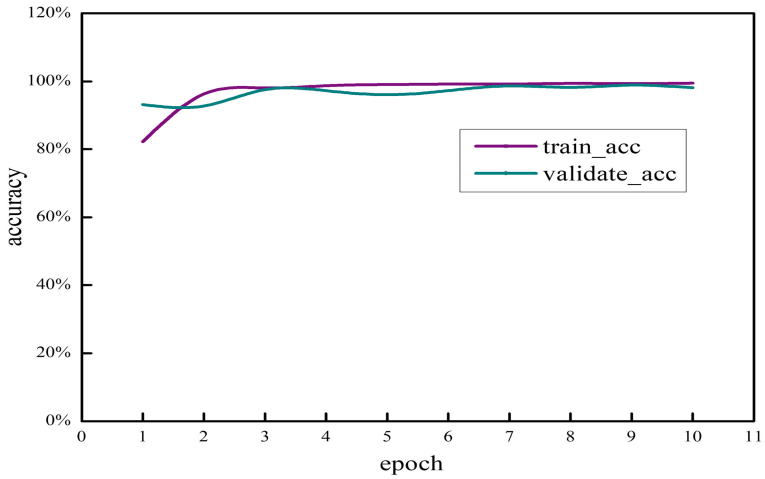
$$False\ alarm\ rate = \frac{FP}{FP + TN} \tag{10}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}$$

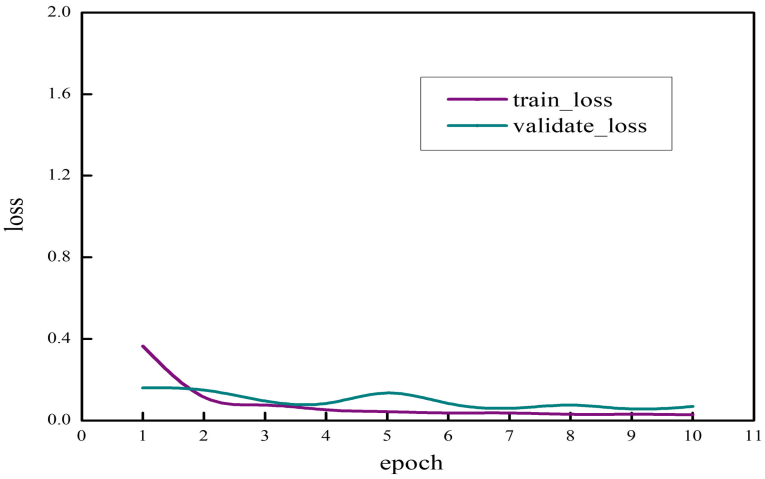
$$F_1\ score = 2 \cdot \frac{precision \cdot detection}{precision + detection} \tag{12}$$

4.3 Results and Discussions

We used batch training methods to train the Bi-LSTM for 10 epochs. The batch size is set as 128 and the validation_split is set as 0.1. We Train on 43966 samples and validate on 4886 samples. The training accuracy and loss and the validation accuracy and loss every one epoch are recorded. The trends of the metrics are



(a) accuracy



(b) loss

Fig. 10. Accuracy and loss in the training stage

presented in Fig. 10. Figure 10(a) shows the accuracy trends, where the orange curve represents the validation accuracy and the dark cyan represents the training accuracy. It shows that after about 7 epochs of training, both the training and validation accuracies have achieved above 98%. Figure 10(b) shows the loss trends, where the orange curve represents the validation loss and the dark cyan represents the training loss. Clearly, both the training and validation losses decrease rapidly towards 0. The trends of accuracy and loss reflect the good capability of the Bi-LSTM.

We evaluate its ability of detecting Web attacks by running the trained Bi-LSTM on test data after 10 epochs of training, detection rate is 98.17%, false alarm rate is 1.40%, test accuracy is 98.35%, precision is 99.00% and F_1 score is 98.58%. This illustrates that with a certain amount of training, the Bi-LSTM has achieved state-of-the-art results in detecting Web attacks, which have both a high detection rate and a low false alarm rate.

Compared with Zhang [14]'s method, our method has achieved better results. Our experimental results show that BL-IDS can greatly improve the accuracy and detection rate while maintaining a low false alarm rate. Our analysis suggests that HTTP requests are more like natural languages, because they can all be considered as a sequence, and there is a temporal relationship between the sequences. So HTTP requests are more suitable to be processed by recurrent neural networks such as Bi-LSTM. However, convolutional neural networks are better at processing image tasks.

5 Conclusion

Exploring a deep learning method to detect Web attacks, which is based on the RNN with the Bi-LSTM. The method can detect Web attacks through inspecting the HTTP request packets. First, studing data preprocessing, which selects useful information from HTTP request packets and produce many word sequences. Second, studing the embedding method used to map words to vectors. Finally, a Bi-LSTM is used to extract features automatically and classify the HTTP request packets to normal or abnormal class. We conducted experiments on the dataset HTTP DATASET CSIC 2010 to evaluate the effectiveness of the method. The results show that the Bi-LSTM can be trained easily and the detection method have a high detection rate and low false alarms in detecting Web attacks.

References

1. Axelsson, S.: Research in intrusion-detection systems: a survey. Technical report 98–17, Department of Computer Engineering, Chalmers University of Technology (1998)
2. Garcia, T.P., Diaz, V.J., Macia, F.G., et al.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(1), 18–28 (2009)
3. OWASP. https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF

4. Lupták, P.: Bypassing Web application firewalls. In: Proceedings of 6th International Scientific Conference on Security and Protection of Information, pp. 79–88 (2011)
5. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
6. Kruegel, C., Vigna, G., Robertson, W.: A multi-model approach to the detection of web-based attacks. Comput. Netw. **48**(5), 717–738 (2005)
7. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidan, R.: N-gram-based detection of new malicious code. In: Proceedings of the 28th Annual International Computer Software and Applications Conference, COMPSAC 2004, vol. 2, pp. 41–42. IEEE (2004)
8. Moh, M., Pininti, S., Doddapaneni, S., Moh, T.S.: Detecting Web attacks using multi-stage log analysis. In: IEEE International Conference on Advanced Computing, pp. 733–738 (2016)
9. Cao, L.C.: Detecting web-based attacks by machine learning. In: 2006 International Conference on Machine Learning and Cybernetics, pp. 2737–2742. IEEE (2006)
10. Cui, J., Long, J., Min, E., Mao, Y.: WEDL-NIDS: improving network intrusion detection using word embedding-based deep learning method. In: Torra, V., Narukawa, Y., Aguiló, I., González-Hidalgo, M. (eds.) MDAI 2018. LNCS (LNAI), vol. 11144, pp. 283–295. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00202-2_23
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
13. Valeur, F., Mutz, D., Vigna, G.: A learning-based approach to the detection of SQL attacks. In: Julisch, K., Kruegel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 123–140. Springer, Heidelberg (2005). https://doi.org/10.1007/11506881_8
14. Zhang, M., Xu, B., Bai, S., Lu, S., Lin, Z.: A deep learning method to detect web attacks using a specially designed CNN. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.-S.M. (eds.) ICONIP 2017. LNCS, vol. 10638, pp. 828–836. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70139-4_84
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
16. HTTP DATASET CSIC 2010. http://www.isi.csic.es/dataset/