Check for updates

Intelligent IoT - Artificial Intelligence for Future Internet of Things - Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2020, Vol. 16(4)
© The Author(s) 2020
DOI: 10.1177/1550147720917019
journals.sagepub.com/home/dsn
SAGE

Network attack detection and visual payload labeling technology based on Seq2Seq architecture with attention mechanism

Fan Shi[1], Pengcheng Zhu[2], Xiangyu Zhou[3], Bintao Yuan[1] and Yong Fang[3]

Abstract

In recent years, Internet of things (IoT) devices are playing an important role in business, education, medical as well as in other fields. Devices connected to the Internet is much more than the number of world population. However, it may face all kinds of attacks from the Internet easily for its accessibility. As we all know, most attacks against IoT devices are based on Web applications. So protecting the security of Web services can effectively improve the situation of IoT ecosystem. Conventional Web attack detection methods highly rely on samples, and artificial intelligence detection results are uninterpretable. Hence, this article introduced a supervised detection algorithm based on benign samples. Seq2Seq algorithm is been chosen and applied to detect malicious web requests. Meanwhile, the attention mechanism is introduced to label the attack payload and highlight labeling abnormal characters. The results of experiments show that on the premise of training a benign sample, the precision of proposed model is 97.02%, and the recall is 97.60%. It explains that the model can detect Web attack requests effectively. Simultaneously, the model can label attack payload visually and make the model "interpretable."

Keywords

Seq2Seq, IoT devices, Web attack detection, autoencoder, attention mechanism, attack visualization

Date received: 3 November 2019; accepted: 6 March 2020

Handling Editor: Kien Nguyen

Introduction

Today portable devices are playing an important role in business, education, and medicine as well as in other fields. IoT devices can provide convenience for human life in auxiliary medical treatment,[1] sleep detection,[2] and activity analysis.[3] Meanwhile, IoT devices may also be used by attackers, leading to privacy disclosure.[4] The IoT devices, which is of easy deployment and scalability, offer a choice instead of traditional desktop programs and greatly facilitate people's daily life and work. IoT devices usually use Web application to provide services to users, so Web attack is also an effective method to attack IoT devices. Therefore, attacks against IoT devices can be launched from the Web application. For its accessibility, it may receive all kinds of attacks from the Internet easily. Meanwhile, its vulnerability is exacerbated due to its distribution and the

[1]College of Electronic Engineering, National University of Defense Technology, Hefei, China
[2]College of Electronics and Information Engineering, Sichuan University, Chengdu, China
[3]College of Cybersecurity, Sichuan University, Chengdu, China

Corresponding author:
Yong Fang, College of Cybersecurity, Sichuan University, Chengdu 610065, China.
Email: yongfangscu@gmail.com

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).

complexity of its configuration. These factors are relevant to the fact that Web attacks are happening with increasing frequency, through which attackers retrieve or change sensitive data or even execute arbitrary code in remote systems.

Most attacks against IoT devices are based on Web applications. To deal with the various attack methods, it has been a trend for security researchers to apply machine-learning and deep-learning to web applications attack detection technique.

A trust-aware probability marking traceback scheme is proposed to locate malicious source quickly.[5] The nodes are marked with different marking probabilities according to its trust which is deduced by trust evaluation. The high marking probability for low trust node can locate malicious source quickly, and the low marking probability for high trust node can reduce the number of marking to improve the network lifetime, so the security and the network lifetime can be improved in this scheme. Wu et.al.[6] proposed a safety detection mechanism based on the analysis of big data. Fuzzy cluster analytical method, game theory, and reinforcement learning are integrated seamlessly to perform the safety detection. The simulation and experimental results show the advantages of this scheme in terms of high efficiency and low error rate.

Adeva and Atxa[7] proposed another attack identify method. This method helps extract metadata from the Weblog, including date, source address, size, type, and so on. Besides, it allows selecting the best feature through the feature assessment, classifying log samples or identifying attacks with Naive Bayes algorithm,[8] k-nearest neighbors (KNN) algorithm,[9] and Rocchio algorithm.[10] The method has a high detection rate of more than 90%. However, it is just a post-test and cannot defend attacks in time. Raghuveer and Chandrasekhar[11] proposed a detection model combining support vector machine (SVM),[12] fuzzy neural network,[13] and K-means.[14] This model clusters and generates various subsets by K-means algorithm, and then trains different neuro-fuzzy models to gain eigenvectors. Rathore et al.[15] designed a cross-site scripting (XSS) classifier based on social networking services (SNS) website, three types of eigenvalues (URL, HTML labels, and SNS) are processed by artificial choice, meanwhile the classifier constructs eigenvectors. Then, 10 machine-learning algorithms, including RF (Random Forest),[16] ADTree (Alternating Decision tree),[17] SVM, LR (Logistic Regression)[18] and so on, will testify and identify whether the eigenvectors are attacks. This method compares each algorithm and obtains the best algorithm model, but it has limitation on scalability and human factors which greatly affect the detection results, as it requires artificial maintenance and obtaining large numbers of eigenvectors artificially. Through Web visit statistics, Yang et al. observed that normal HTTP requests were a majority, and the behavior patterns were similar, while malicious requests were a minority and the behavior patterns were changeable.[19] They proposed an unsupervised algorithm based on text clustering to distinguish normal requests from malicious requests. Rather, it proved to be in high detection rate and low false alarm rate. Zhang et al. picked up upper 300 bytes characters in Web communication traffic through statistical analysis to Webshell traffic. The sequence vector was generated based on American Standard Code for Information Interchange (ASCII).[20] Then CNN (convolutional neural networks)[21] and LSTM (long short-term memory)[22] trained sequence vectors to build a model and classify sample data. This method obtained rather good results with a 98.2% detection rate and 97.84% recall rate.

The above detection methods affect well since data sets have been given. However, there are still some problems that need resolving:

1. The lack of label data: There are numerous normal request samples while there are few variegated attack samples in real environment, which causes obstacles to the model's learning and training.
2. The lack of sample classes: In the stage of training, if there are only SQL injection and XSS attacks in the sample data sets it is hard to identify command executions or new Payload attacks in real environment. Besides, Web applications run by different users vary greatly. Even SQL injection has numerous forms. Obviously, we cannot be sure to use a data collected in the past to train a model that can detect unknown attacks. It is easy to understand that the results will be deadly different in the experimental environment from the real environment.
3. The interpretability of the results: If the model identifies SQL injection, security researchers can find out the exact location of the attack payload so that they can maintain the Web applications consciously. But common Web maintainers may not understand the significance of the alarm. Even though they constrain the attacks at that very moment, they still cannot repair the Web applications. Because of that, the security risk still exists.

As services provided by IoT devices are often subject to Web attacks, to improve the security of IoT devices, an attack detection model based on Seq2Seq[23] to implement the shortage of current Web attack detection technologies is proposed in this article. This model helps acquire a great many of normal samples, identify kinds

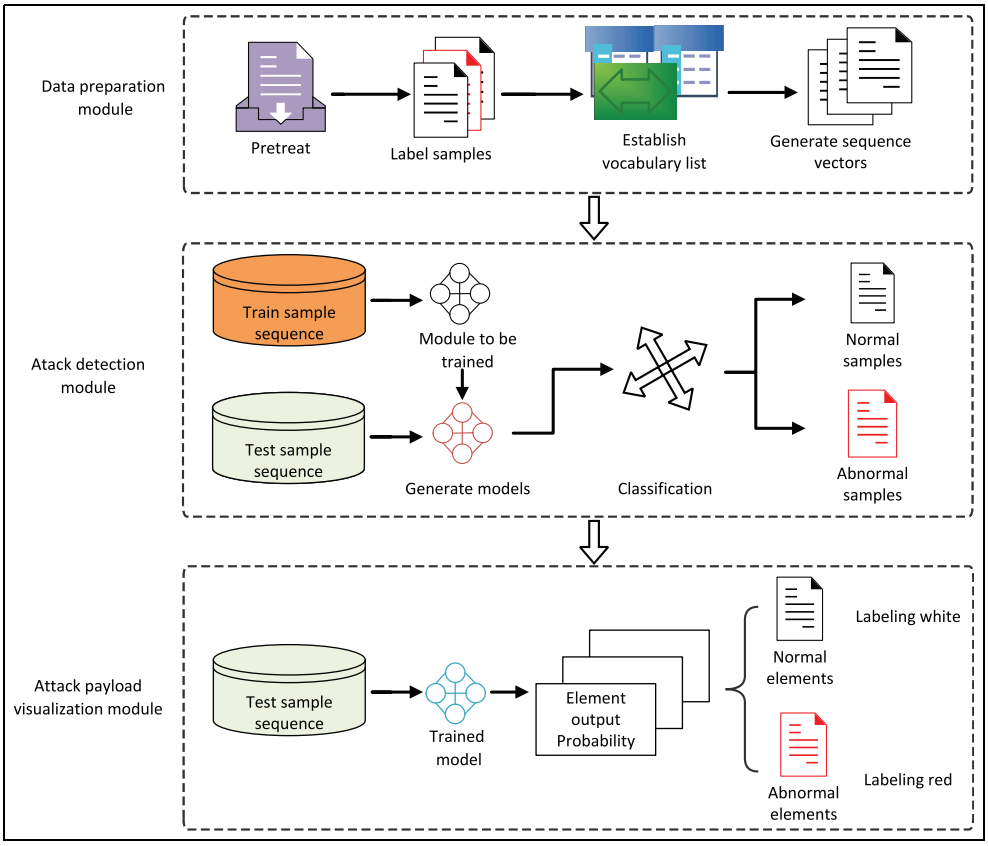

Figure 1. Diagram of detection model framework.

of Web attacks efficiently, and locate attack payload timely. We summarize the major contributions as following:

1. We propose a visual payload labeling model to detect network attack. Under the premise of using only benign training samples, our model has good precision and recall.
2. As our model relies on comparing predicted values and thresholds to classify benign and malicious Web requests, it can identify whether a Web request is malicious, rather than defend against a specific type of Web attacks.
3. Our model not only distinguishes normal requests from attack requests but also interprets the detection results by visually labeling the attack payloads. In the stage of encoding, we encode request samples from HTTP through the Bi-LSTM algorithm and maintain the context semantics in the request. In the decoding stage, we introduce the attention mechanism, calculate the probability distribution of each character in the sequence vector, and mark the exact location of the attack payload. The detect results of our model are interpretable. Website maintainers are able to locate the attack payload swiftly, repair security risks in time, and protect the data security of enterprises or organizations.

Web attack detection model based on Seq2Seq

Detection model framework

Figure 1 presents the whole framework of the Web attack detection model based on Seq2Seq. The model is mainly divided into three modules: data preparation module, attack detection module, and attack payload visualization module. In the data preparation module, pretreating original HTTP request samples, establishing vocabulary, generating sequence vectors that meet the model's input requirements happend in sequence. In the attack detection module, the main task is to construct and train the attack detection model as well as testify and classify the test sample sets. In the attack payload visualization module, the attack payload is visually labeled, and normal elements (characters) are labeled as white but abnormal elements (characters) are labeled as red.

" ": 32,"!": 33,""": 34,"#": 35,"$": 36,"%": 37,"&": 38,"'": 39,"(": 40,")": 41,"*": 42,"+": 43,",": 44,"-": 45,".": 46,"/": 47,"0": 48,"1": 49,"2": 50,"3": 51,"4": 52,"5": 53,"6": 54,"7": 55,"8": 56,"9": 57,":": 58,";": 59,"<": 60,"=": 61,">": 62,"?": 63,"@": 64,"A": 65,"B": 66,"C": 67,"D": 68,"E": 69,"F": 70,"G": 71,"H": 72,"I": 73,"J": 74,"K": 75,"L": 76,"M": 77,"N": 78,"O": 79,"P": 80,"Q": 81,"R": 82,"S": 83,"T": 84,"U": 85,"V": 86,"W": 87,"X": 88,"Y": 89,"Z": 90,"[": 91,"\": 92,"]": 93,"^": 94,"_": 95,"`": 96,"a": 97,"b": 98,"c": 99,"d": 100,"e": 101,"f": 102,"g": 103,"h": 104,"i": 105,"j": 106,"k": 107,"l": 108,"m": 109,"n": 110,"o": 111,"p": 112,"q": 113,"r": 114,"s": 115,"t": 116,"u": 117,"v": 118,"w": 119,"x": 120,"y": 121,"z": 122,"{": 123,"|": 124,"}": 125, "<PAD>": 0,"<GO>": 1,"<EOS>": 2,"<UNK>": 4,"\r":5,"\n":6,"\t":7

Figure 2. Normal request character embedded vocabulary.

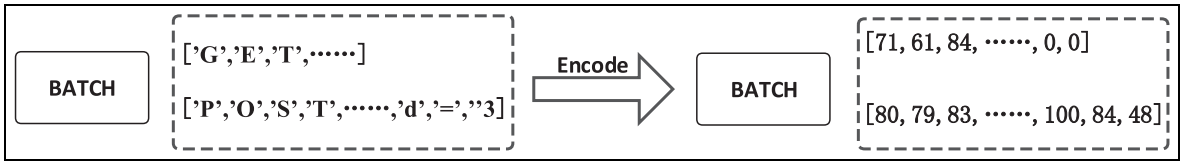

Figure 3. Sample encode example.

Data preparation module

Sample labeling. Sample cleansing and labeling play a key role in machine learning, as samples' quality determines the quality of model training and the accuracy of subsequent detections. Through the observation of the data sets, we could conclude that the data of benign samples accords with expectations while mislabeled many malicious samples, and we found that the length of most wrongly labeled data requests is less than 20 bits. So, the first step is to delete all the POST and GET request data which is less than 20 bits, and then label the samples manually to ensure all our training samples are labeled correctly. Though we would delete some data wrongly, the sample data labeling is more accurate, guaranteeing the accuracy of the experiments. After cleansing, we store rest of the samples, labeling abnormal HTTP requests as "malicious" and normal HTTP requests as "benign."

Establish vocabulary based on ASCII. Paper[20] applied the Webshell traffic detection technology which is based on deep-learning. Similar to the method introduced in that paper, the model in this article constructs sequence vectors by embedding characters. The first step is to establish vocabulary with the following steps:

1. We store the visible characters in the vocabulary, and the index number of each character is its corresponding ASCII code.
2. Per the autoencoder model introduced in section "Experiment and assessment," we input <GO> first at the decoding sequence, index number 0, and output <EOS> last at the sequence, index number 2.
3. Setting the characters as <UNK> which are not in the vocabulary or we cannot identify, index number "4."
4. The length of ordered column vectors needs to be filled into the same length to meet the requirements of Bi-LSTM. <PAD> is the filler word, index number "0."
5. Link break "/r," tab "t" is added to the vocabulary. Figure 2 shows the final vocabulary. Figure 3 elaborates sequence vectors after we encoding samples in one batch in training.

Attack detection module based on Seq2Seq

Seq2Seq model framework. Figure 4 refers to Seq2Seq (we choose the Seq2Seq framework because its output length is uncertain, which we do not need modify the input vector) framework. As the figure shows, X = [X1, X2, X3, X4] refers to input sequences, and Y = [Y1, Y2, Y3] refers to output sequences. Encoder and decoder can be various neural network models or their combinations. We believe that the payload of a Web attack has a context, and Bi-LSTM can better capture the bidirectional semantic dependency, so after encoding with Bi-LSTM, this context can be expressed

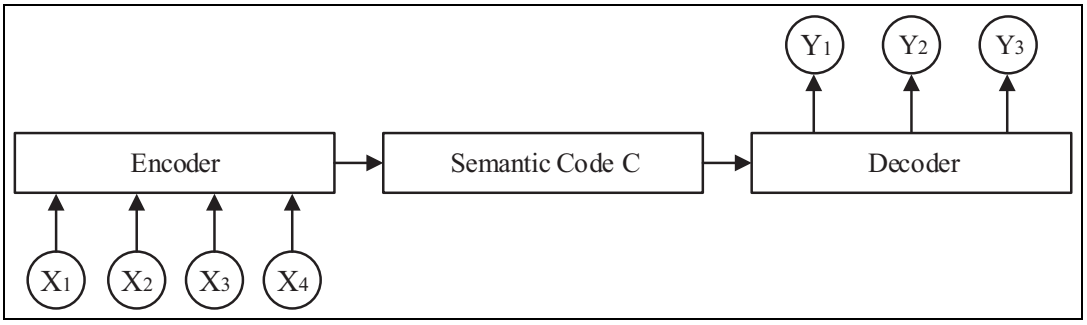

Figure 4. Basic framework of Seq2Seq model.

in the vector. Semantic Coding C refers to the encoding value of sequence X.

1. Encoder

At the stage of encoding, for the Encoder is Bi-LSTM, its hidden layer output is the splicing of farward and reverse LSTM hidden layer ouput. Shown as follows

$$H_t = [h_t, \partial h_t] \tag{1}$$

$$h_t = f(h_{t-1}, x_t) \tag{2}$$

$$h_t^{'} = f(h_{t-1}, x_t) \tag{3}$$

f is the encoding function of Bi-LSTM, h_t is the output of the forward LSTM hidden layer, and h_t' is the output of the reverse LSTM hidden layer. For semantic coding C, the output information of Encoder's hidden layer is generally aggregated to obtain the semantic vector of the middle layer

$$C = q[H_1, H_2, H_3, ..., H_t] \tag{4}$$

A common simple method is to use the hidden layer output of the last moment as the semantic vector C, that is

$$C = q[H_1, H_2, H_3, ..., H_t] = H_t \tag{5}$$

2. Decoder

The decoding stage can be regarded as the inverse process of encoding. For the output Y_t at a certain time, it is predicted by the output sequence $Y_1, Y_2, Y_3, ...Y_{t-1}$ and semantic vector C, as shown in formula (6)

$$Y_t = argmaxP(Y_t) = \sum_{t=1}^{T} p(Y_t|\{Y_1, Y_2, Y_3, ...Y_t\}, C) \tag{6}$$

it can be abbreviated as

$$Y_t = g(Y_t|\{Y_1, Y_2, Y_3, ...Y_t\}, C) \tag{7}$$

For the model that decoder is Bi-LSTM, Y_t is determined by the output of Y_{t-1} at the last moment, the output of hidden layer at the current moment and the semantic vector C. The above formula can be abbreviated as formula (8)

$$Y_t = g(Y_{t-1}, H_t^{'}, C) \tag{8}$$

Attack detection algorithm based on measuring the loss of model. The last section introduced the basic framework of Seq2Seq. Seq2Seq needs modifying before it is applied to detect attacks. In Figure 5, we take training samples as input and output of the model at the same time, and this model is also called autoencoder. This model is almost the same as the Seq2Seq model diagram in Figure 3. The main difference is that the output layer also uses the same data as the input layer. It should be noted that in the decoding stage, the first input of the sequence is replaced by "<GO>," and the last output of the sequence is replaced by "<EOS>."

To train only with positive samples and process attack detection, this article designed an attack detection algorithm based on measuring the loss of a model. The procedures are as follows:

1. To gain a model with slight loss by training a great many of benign sample sets.
2. To predict benign samples in test sets. Under normal circumstance, every sequence has one predictive value with rather low loss. The loss of all the sequences is counted and recorded as *total_loss*

$$total\_loss = [Loss_1, Loss_2, ..., Loss_T] \tag{9}$$

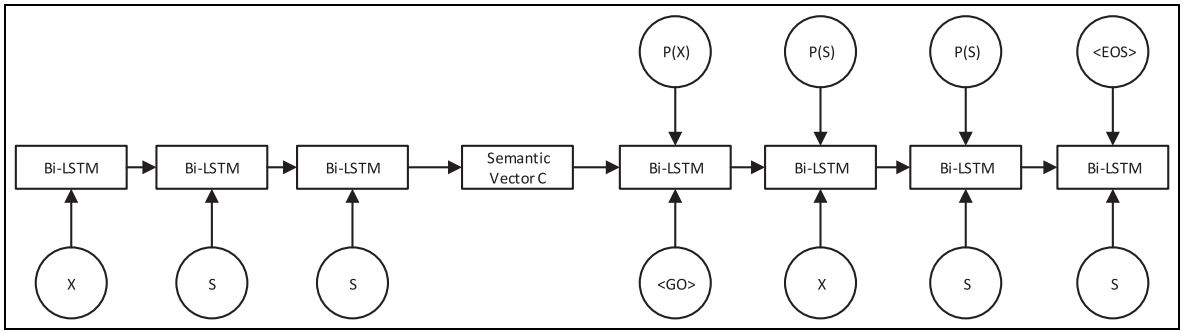

Figure 5. Basic framework of Seq2Seq model (Autoencoder).

3. Calculate the mean and standard deviation of *total_loss* in equation (2), and calculate the threshold using the following formula

$$threshold = mean(total\_loss) + C * std(total\_loss) \quad (10)$$

In the above formula, *mean* refers to calculate mean value, and *std* refers to calculate the standard deviation. C is a constant, and we need to adjust it in experiments, so that *threshold* can gradually approach the optimal threshold. Generally speaking, C should ensure that the threshold value is greater than the maximum value of *Loss.Max* of the test set *Loss*.

4. The model predicted benign samples and malicious samples at the same time. If the *Loss*>*threshold*, the malicious samples are judged; otherwise, the *Loss*<*threshold* of the sequence is the normal samples.

Attack payload visualization module based on attention mechanism

Seq2Seq model with attention mechanism. To solve the problem that cannot explain the results of the conventional detection model, this section will optimize the Seq2Seq model using the attention mechanism and mark the specific location of attack payload using the characteristics of this mechanism, to realize the visualization function of attack payload. The optimized model is shown in Figure 6.

1. Encoder

After leading into the attention mechanism, the semantic vector C is obtained by weighted averaging the output H of encoder's hidden layer, as follows

$$C_i = \sum_{j=1}^{T} p(a_{ij}H_j) \quad (11)$$

a_{ij} represents the corresponding weight of each hidden layer and is calculated by the following formula

$$a_{ij} = softmax(e_{ij}) \quad (12)$$

e_{ij} is a score calculated by the output H_i of the encoder's hidden layer and the output H_j' of the decoder's hidden layer. The score is calculated by the following formula

$$e_{ij} = score(H_i, H_j^{'}) \quad (13)$$

For the score function, Luong et al.[24] defines the following three definitions, which can be selected according to the different problems

$$score(H_i, H_j^{'}) = \begin{cases} H_i^T H_j^{'} \\ H_i^T W_a H_j^{'} \\ V_a tanh(W_a[H_i^T; H_j^{'}]) \end{cases} \quad (14)$$

2. Decoder

The decoding stage is determined by the current time semantic vector C_t and the output $H_t^{'}$ of the decoder's hidden layer. First, we contacted the two vectors, and use *tanh* as the activation function

$$H_t^* = tanh(W_c[C_t; H_t^{'}]) \quad (15)$$

Finally, the predicted output Y_t is calculated

$$Y_t = argmaxP(Y_t) = softmax(W_c, H_t^*) \quad (16)$$

We should note that the output Y_t at this time is a probabilistic sequence. By looking up the maximum probability value in the sequence, the corresponding word is retrieved from the vocabulary list and decoded.

Attack payload labeling principle based on attention mechanism. Formula (15) shows that the output H_t^* of the attention layer is determined by semantic vector C_t and $H_t^{'}$, the output of the Decoder's hidden layer. At

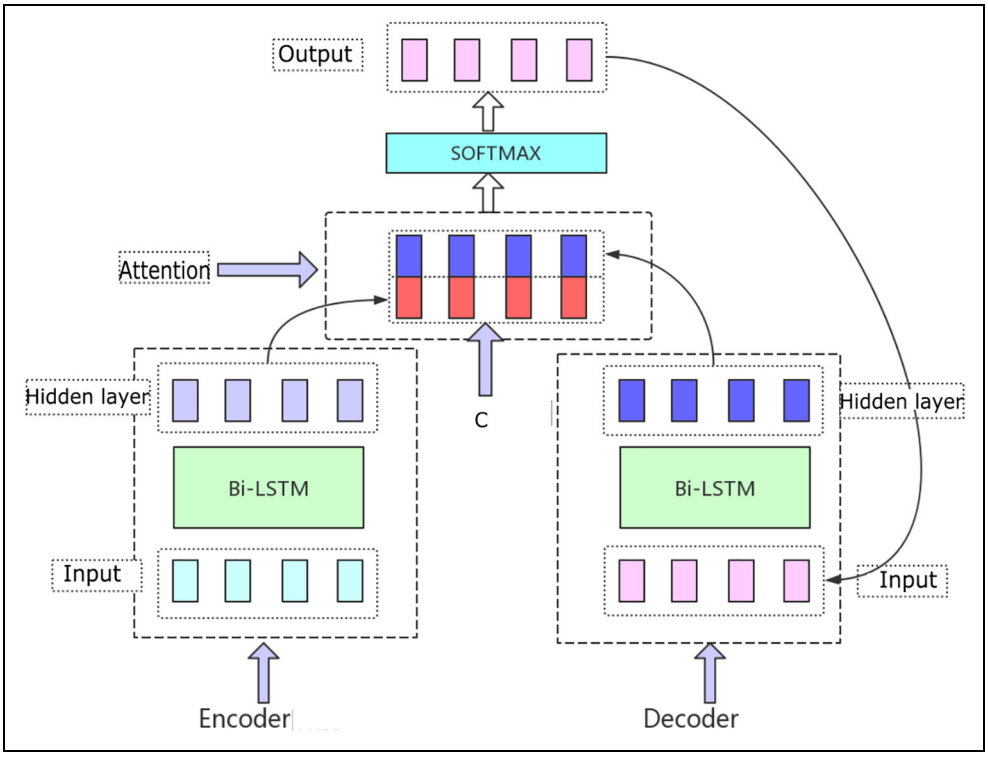

Figure 6. Seq2Seq model with attention mechanism.

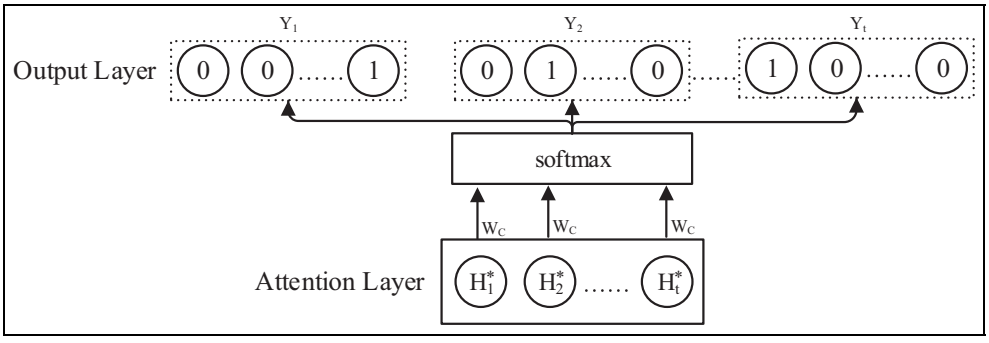

Figure 7. Attention layer and output layer diagram.

this time, the semantic vector C_t represents the weight of the current input compared to the output of the model, which is similar to the human attention mechanism. It is the focus of visual attention with a large weight, and on the contrary, it may be low-value information with a low weight.

Assuming that the input of X_t is the No.i element in the vocabulary at a certain time, the output of Y_t at the current time is a probabilistic sequence of sum 1, which is as long as the vocabulary and the sum of them is 1, as shown in Figure 7 and Formula (16). On the premise of correct prediction, the No.i element of the sequence should be the maximum value of the sequence and far greater than the value of other elements. (Figure 6 is a simple demonstration. The No.i element of the probabilistic sequence is set to 1 and the rest is set to 0.)

Based on this conclusion, the following steps can be taken to optimize the model:

1. The test samples are predicted by the trained model, receive the output probabilistic sequence

Table 1. Data set.

Classification of experiments	Training samples		Testing samples		Detection samples		Total
	Benign	Malicious	Benign	Malicious	Benign	Malicious	
Model in this article	16,264	0	0	0	4067	4067	24,398
Classification threshold detection and computation	0	0	4880	0	0	0	4880
Threshold calculation of abnormal elements	0	0	1001	1001	0	0	2002
Comparative experiment	16,264	12,176	0	0	4067	4067	36,574

$$Y_{ij} = [\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, ..., \alpha_{it}] \quad (17)$$

In the formula, Y_{ij} refers to the No.i sequence, the No.j element in the vocabulary, T is the length of the vocabulary, recording the current value of α_{ij}.

2. Count all outputs of samples, set as α

$$alpha = [\alpha_{ij}] \quad (18)$$

Calculate the mean value and standard deviation of α, and use the following formula to calculate the threshold. In the formula, the constant C is a constant to be determined, mean value is calculated by *mean*, and standard deviation is calculated by *std*

$$threshold = mean(alpha) - C * std(alpha) \quad (19)$$

3. By adjusting the constant C, make sure the threshold value is guaranteed to be less than the minimum weight of benign samples in the test set, and greater than the maximum weight of malicious samples, such as Formula (20). Meanwhile, it is necessary to observe whether the sample labeling conforms to objective facts. If it does, the threshold value should be selected; otherwise, it will continue to adjust

$$malicious_{max} < threshold \leqslant benign_{min} \quad (20)$$

4. If a sequence in the test set is checked by the model, the model predicts the No.j element $a_{ij} < threshold$ in the probabilistic sequence of Y_{ij}, it indicates that Y_{ij} is abnormal and labeled red; otherwise, if $a_{ij} > threshold$, it indicates that Y_{ij} is normal and labeled white.

Experiment and assessment

Data set

This article applied HTTP DATASET CSIC 2010 data set to to do experiments and make analysis.[25] After processing, we stored 20,331 pieces of benign samples and 16,243 malicious samples. According to the detection method introduced in this article, we divide data sets into three parts: training samples, testing samples, and detection samples. Among them, we take 30% benign samples randomly to do threshold test training, and 1001 benign samples and 1001 malicious samples randomly to do abnormal element threshold test. Away from that, this model only adopts benign samples in training, but in comparative experiments, it adopts benign samples and malicious samples simultaneously. The specific allocation of data sets is shown in Table 1.

Environment for experiment

The model introduced in this article is mainly developed under Windows system. The code involved in the experiment is mainly based on Python's tensorflow framework.[26] Function *static_bidirectional_rnn* in tensorflow is applied to realize encoder of Bi-LSTM algorithm. Function Seq2Seq in tensorflow is applied to realize encoder with attention mechanism. Python Scikit-learn toolkit helps to realize assessment for model.[27] Detailed configurations are shown in Table 2.

Experiment process

Classification threshold parameter optimization. In sections "Attack detection module based on Seq2Seq" and "Attack payload visualization module based on attention mechanism," the calculation methods of threshold value of model classification and threshold value of exceptional determination have been introduced in detail, but the formula cannot directly calculate the final threshold value. Further experimental tests are necessary to get the optimal threshold value. Formula (10) shows that constant C needs to be adjusted to obtain a reasonable threshold value to achieve the goal of sample classification. We tested the accuracy change with a constant C from 1 to 7 in steps of 2, and specifically tested the accuracy with a constant 0. The relationship between threshold value and accuracy is shown in Table 3.

It is understandable that the higher the threshold is, the higher the accuracy of benign samples is, but the model also needs to detect malicious samples, so while

Table 2. Hardware and software configuration of experimental environment.

Operating system	Microsoft Windows 10 Build 17134 Professional edition, 64-bit operating system
System configuration	CPU: Intel i7-7700, Memory: 8GB, Hard Disk: 1TB GPU: NVIDIA GeForce GTX 1060, Display Memory:6GB
The Python standard library and version	Python:3.6.2 tensorflow-gpu == 1.12.0 numpy == 1.16.0 scikit-learn == 0.19.2 matplotlib == 2.2.2 colorama == 0.4.1

Table 3. Threshold test results.

Number	Constant C	Threshold	Accuracy (%)
1	0	0.076447	72.57
2	1	0.215574	94.27
3	3	0.493829	99.20
4	5	0.772084	99.54
5	7	1.050339	99.58

ensuring the accuracy, the smaller the threshold, the more consistent with the classification standard of the model. As shown in the table above, when the constant C is 5 and 7, the accuracy no longer increases significantly, and the threshold value is 0.772084.

Threshold parameter optimization of abnormal elements. Because we cannot quantify the threshold value of the classification of abnormal elements accuracy, we only calculate an estimated value by Formula (19): $threshold = mean(alpha) - C * std(alpha)$. If the value meets the expectation, it will be adopted, conversely, then adjusted. According to the statistics of 1001 benign samples and 1001 malicious samples, $mean(alpha) = 0.67052$ and $std(alpha) = 0.4568767$ were obtained. Threshold adjustment calculations are as follows

1. Initial value $C = 0$, step size $= 0.1$ and maximum value $= 1.5$;
2. Calculate threshold value by formula (19);
3. Ten malicious samples and 10 normal samples were printed randomly to observe whether to label the labels of attack payload;
4. If it does not meet the expectation, repeat (1) to (3) until it meets the expectation, and store the current threshold value.

After many rounds of experiments, we set the threshold value to 0.076589 (constant $C = 1.3$), when the output meets the target of labeling attack payload. Figure 8 is an example of labeling attack payloads.

Experiment indicators

To get better evaluation of the attack detection model based on attention mechanism and Bi-LSTM algorithm, the experimental results will be evaluated using Confusion Matrix. The obfuscation matrix, also known as Error Matrix, can be used to visually evaluate the performance of classification model algorithms, as shown in Table 4.

```
1. Prediction: 1
2. Loss: 8.41622
3. POST /vulnbank/onlinebank/api.php?xml HTTP/1.1
4. Host: 10.0.212.25
5. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0
6. Accept: application/json, text/javascript, */*; q=0.01
7. Content-Type: text/xml
8. Content-Length: 945
9. Connection: close
10. <?xml version="1.0" encoding="utf-8lz [<!ENTITY lol "lol"><!ELEMENT login
(#PCDATA)><!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;"><!ENTITY lol2
"&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;"><!ENTITY lol3
"&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;"><!ENTITY lol4
"&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;"><!ENTITY lol5
"&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;"><!ENTITY lol6
"&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;"><!ENTITY lol7
"&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;"><!ENTITY lol8
"&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;"><!ENTITY lol9
"&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">]><api><type>user</type><action>
forgotpass</action><username>j.doe</username><password>&lol9;</password><code>000
</code></api><EOS>
```

Figure 8. An example of labeling attack payloads.

Table 4. Confusion Matrix.

	Malicious requests	Normal requests
Malicious requests	TP	FP
Normal requests	FN	TN

TP: true positive; FP: false positive; FN: false negative; TN: true negative.

The matrix has the following four categories of indicators:

1. True positive (TP): An attack request and the result of model judgment is also a sample of the attack request.
2. False positive (FP): Samples which are real normal requests but whose model judgment results are attack requests.
3. True negative (TN): Samples that are real normal requests and the results of model judgment are also normal requests.
4. False negative (FN): Samples that are attacking requests but the model judges the results as normal requests.

Based on the above definition and the detection of attack behaviors in this article, we can consider it as a binary classification, which can deduce the performance indicators of this model

$$Precision : P = \frac{TP}{TP + FP} \tag{21}$$

$$Recall : R = \frac{TP}{TP + FN} \tag{22}$$

$$F_1 score : F_1 = 2 \times \frac{P \times R}{P + R} \tag{23}$$

The precision reflects the proportion of real malicious requests in the sample in which the results of model judgment are malicious requests; the recall reflects the proportion of malicious requests correctly identified by the model, and the F1 score is the harmonic mean of the precision rate and recall rate.

Results

To make the experiment's objective, we compare results from the model introduced in this article with the other four attack detection models:

1. SVM: Characteristic construction feature vectors are extracted artificially, and then the samples are classified by SVM algorithm.
2. TF-IDF_RF: Word frequency vectors are extracted from samples by TF-IDF, and then the samples are classified by Random Forest.
3. Word2vec_MLP: The samples are segmented artificially, and then word vectors with semantics are constructed by Word2vec. At last, the samples are classified by MLP algorithm.
4. Character_CNN: Sequence vectors are constructed by character embedding. Finally, samples are classified by CNN algorithm.

Table 5 shows that they use different methods to extract features: SVM extracts features artificially, Word2vec_MLP needs semantic model constructed artificially, and the rest three models extract features with the help of algorithms. As for the requirements toward samples, the Attention_Bi-LSTM model in this article only needs to train positive samples, but the rest four models all need samples with positive and negative labels. For the interpretability of results, only the model introduced in this article achieves attack visualization.

The comparative results of experimental performance indexes are shown in Table 6. Character_CNN has a precision rate of 99.48%, but the recall rate is only 94.66%, indicating that the model has a lower false positive rate (FPR) but a higher false negative rate. Our model does not have the precision accuracy, but the recall rate is 97.60%, and the F1 value is 97.31%, means that our model has better comprehensive ability. The attention_Bi-LSTM model performs better in the aspect of precision, recall, F1 value than SVM, TF-IDF_R, and Word2vec_MLP. Although Attention_Bi-LSTM has lower precision than Character_CNN, it does better in recalling, and it has a higher F1 value. In summary, on the premise of only positive training samples, Attention_Bi-LSTM performs best in classification. Character_CNN and

Table 5. Comparison of model characteristics.

Detection models	Feature extraction method	Sample requirements	Visualization
SVM	Artificial extraction	Positive and negative samples	No
TF-IDF_RF	Algorithm	Positive and negative samples	No
Word2vec_MLP	Artificial segmentation	Positive and negative samples	No
Character_CNN	Algorithm	Positive and negative samples	No
Attention_Bi-LSTM	Algorithm	Positive samples	Yes

Table 6. Model performance indicators.

Detection models	Precision (%)	Recall (%)	F1 (%)
SVM	92.07	94.95	93.49
TF-IDF_RF	93.81	89.12	91.41
Word2vec_MLP	96.28	96.29	96.29
Character_CNN	99.48	94.66	97.01
Attention_Bi-LSTM	97.02	97.60	97.31

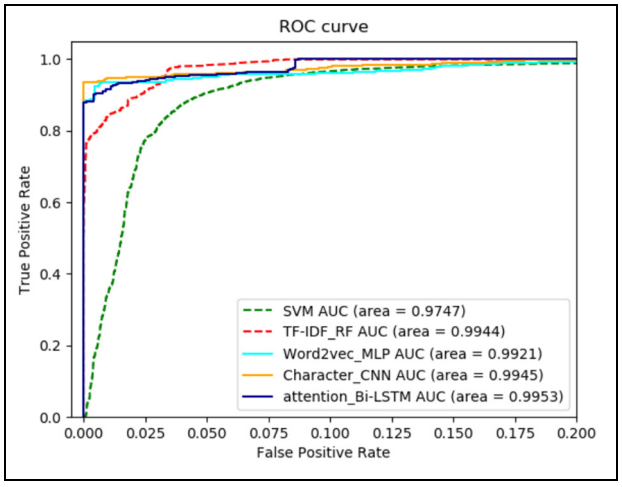

Figure 9. Comparison of ROC curves of different models.

Word2vec_MLP are good. The performance of TF-IDF_RF and SVM is worse.

Figure 9 shows a comparison of the receiver operating characteristic (ROC) curve of five models. The Attention_Bi-LSTM model achieved the best true positive rate (TPR) at smaller FPR, proving that our model has better accuracy than other models. In terms of the degree of automation of the model, our model has higher accuracy using only benign samples for training, and do not need extract features artificially. Although Character_CNN has higher precision than Attention_Bi-LSTM, our model is better than the other four models in terms of construction, training, and accuracy.

Conclusion

Based on experimental data sets, tests on Attention_Bi-LSTM, SVM, TF-IDF_RF, Word2vec_MLP, Character_CNN are processed. The results indicate that SVM and TF-IDF_RF have relatively low detection rate; their precision is 92.07% and 93.81%, respectively; their recall is 94.95% and 89.12%, respectively. The detection and recall of Word2vec_MLP are of average, 96.28% and 96.29%, respectively.

That means extracting word vectors by Word2vec can maintain samples' semantics and make classification as well. The precision of Character_CNN reaches 99.48%, but the recall is 94.66%. It shows that Character_CNN has a high false alarm rate. The precision, recall, and F1 values of Attention_Bi-LSTM are as high as 97.02%, 97.60%, and 97.31%, respectively. Also, Attention_Bi-LSTM has the largest AUC (the area under the ROC curve). It shows that on the premise of benign training samples alone, this model can detect attack requests effectively and has rather high precision and recall. Besides, its exclusive function of labeling attack payload helps to achieve attack visualization.

However, the model has some shortcomings. The model constructs sequence vectors by using character embedding. Although it shortcuts the steps of artificial word segmentation and feature extraction, it enlarges calculation. There are only around 20,000 data sets, but the training time is more than 10 h. We will consider adopting the embedding method of N-gram to process experiments or improve hardware resources of experiments.

Acknowledgements

The authors thank anonymous reviewers and editors for providing helpful comments on earlier drafts of the manuscript.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by National Key Research and Development Program (2016YFE0206700 and 2018YFB0804503) and the Fundamental Research Funds for the Central Universities.

ORCID iD

Yong Fang https://orcid.org/0000-0003-0708-1686

References

1. Gu Y, Wang Y, Liu Z, et al. Sleepguardian: an RF-based healthcare system guarding your sleep from afar, 2019, arXiv:1908.06171v1.
2. Gu Y, Zhang Y, Li J, et al. Sleepy: wireless channel data driven sleep monitoring via commodity WiFi devices. *IEEE T Big Data*. Epub ahead of print 28 June 2018. DOI: 10.1109/TBDATA.2018.2851201.
3. Gu Y, Ren F and Li J. Paws: passive human activity recognition based on wifi ambient signals. *IEEE Internet Thing J* 2015; 3(5): 796–805.
4. Gu Y, Zhang X, Li C, et al. Your WiFi knows how you behave: leveraging WiFi channel data for behavior

analysis. In: *Proceedings of the 2018 IEEE global communications conference (GLOBECOM)*, Abu Dhabi, UAE, 9–13 December 2018, pp.1–6. New York: IEEE.
5. Liu X, Dong M, Ota K, et al. Trace malicious source to guarantee cyber security for mass monitor critical infrastructure. *J Comput Syst Sci* 2018; 98: 1–26.
6. Wu J, Ota K, Dong M, et al. Big data analysis-based security situational awareness for smart grid. *IEEE T Big Data* 2016; 4(3): 408–417.
7. Adeva JJG and Atxa JMP. Intrusion detection in web applications using text mining. *Eng Appl Artif Intell* 2007; 20(4): 555–566.
8. Tang B, He H, Baggenstoss PM, et al. A bayesian classification approach using class-specific features for text categorization. *IEEE Trans Knowl Data Eng* 2016; 28(6): 1602–1606.
9. Li D, Zhang B and Li C. A feature-scaling-based -nearest neighbor algorithm for indoor positioning systems. *IEEE Internet Thing J* 2015; 3(4): 590–597.
10. Ding Q, Zhang J, Wang J, et al. Based on knn and rocchio improved text classification technology. *Autom Instrum* 2017; 8: 41.
11. Chandrasekhar AM and Raghuveer K. Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers. In: *Proceedings of the international conference on computer communication and informatics*, Coimbatore, India, 4–6 January 2013, pp.1–7. New York: IEEE.
12. Suykens JAK and Vandewalle J. Least squares support vector machine classifiers. *Neural Proces Lett* 1999; 9(3): 293–300.
13. Lin CT, George Lee CS, Lin CT, et al. *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*, vol. 205. Upper Saddle River NJ: Prentice Hall, 1996.
14. Krishna K and Murty NM. Genetic k-means algorithm. *IEEE T Syst Man Cyb: Part B* 1999; 29(3): 433–439.
15. Rathore S, Sharma PK and Park JH. Xssclassifier: an efficient XSS attack detection approach based on machine learning classifier on SNSs. *J Inform Process Syst* 2017; 13(4): 1014–1028.
16. Breiman L. Random forests. *Machine Learn* 2001; 45(1): 5–32.
17. Freund Y and Mason L. The alternating decision tree learning algorithm. In: *Proceedings of the sixteenth international conference on machine learning, ICML '99*, Bled, Slovenia, 27-30 June 1999, pp.124–133. New York: ACM.
18. Castilla E, Martín N and Pardo L. A logistic regression analysis approach for sample survey data based on Phi-divergence measures. In: Gil E, Gil E, Gil J, et al. (eds) *Mathematics of the uncertain*. New York: Springer, 2018, pp.465–474.
19. Yang X, Wei LI, Sun M, et al. Web attack detection method on the basis of text clustering. *CAAI Trans Intel Syst* 2014; 9: 40–46.
20. Zhang H, Guan H, Yan H, et al. Webshell traffic detection with character-level features based on deep learning. *IEEE Access* 2018; 6: 75268–75277.
21. Yandong LI, Hao Z and Lei H. Survey of convolutional neural network. *J Comput Appl* 2016; 36: 2508–2515.
22. Greff K, Srivastava RK, Koutník J, et al. LSTM: a search space odyssey. *IEEE T Neural Netw Learn Syst* 2016; 28(10): 2222–2232.
23. Google. seq2seq, 2017, https://google.github.io/seq2seq/
24. Luong MT, Pham H and Manning CD. Effective approaches to attention-based neural machine translation, 2015, arXiv:1508.04025v5.
25. Giménez CT, Villegas AP and Marañón GA. Http data set CSIC 2010. *Information Security Institute of CSIC (Spanish Research National Council)*, 2010, https://www.impactcybertrust.org/dataset_view?idDataset = 940
26. Abadi M, Barham P, Chen J, et al. Tensorflow: a system for large-scale machine learning. In: *Proceedings of the 12th USENIX symposium on operating systems design and implementation (OSDI 16)*, Savannah, GA, 2–4 November 2016, pp.265–283. New York: ACM.
27. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. *J Machine Learn Res* 2011; 12: 2825–2830.