Received November 12, 2019, accepted December 7, 2019, date of publication December 16, 2019, date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2959950

Webshell Detection Based on the Word Attention Mechanism

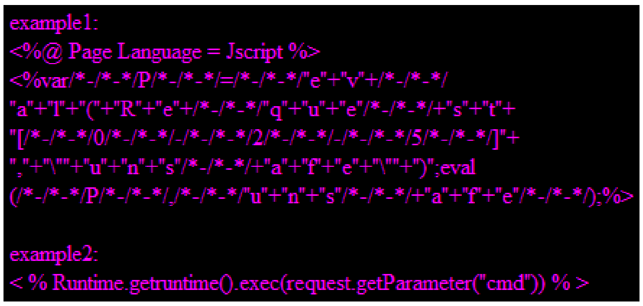
TINGTING LI, CHUNHUI REN, YUSHENG FU, JIE XU, (Member, IEEE), JINHONG GUO, AND XINYU CHEN
School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
Corresponding author: Yusheng Fu (yushengf@uestc.edu.cn)

ABSTRACT Webshell is a backdoor web page-based program. Malicious attackers obtain some privileges through the Webshell so as to realize the operation and control of the website. However, due to confusion coding technology, Webshell detection becomes difficult. This paper presents a Webshell detection model based on the word attention mechanism. In the model, we mainly focus on intra-line word association. After using Word2vec to vectorize the words, we use GRU (Gated Recursive Unit) and the attention mechanism to train and detect the samples. The experimental results show that the model has a high detection rate and low loss function.

INDEX TERMS Webshell, text vectorization, GRU, attention mechanism.

I. INTRODUCTION

Webshell is a web page program written in ASP, PHP, JSP, CGI, or other web scripting languages. Users can use web pages to upload files, view databases, and execute operating system commands through browsers. It can be used as a backdoor because malicious users can also launch web attacks through malicious web pages [1]. To avoid detection, attackers often use methods of confusing encryption to hide Webshells, such as: camouflage, which changes the name to ''ordinary''; encryption, which encrypts the execution code first and decrypts before executing; embellishing, which inserts a large number of useless random strings into the intermediate code; and polymorphism, which makes some judgments before executing and only executing when matching conditions [2]. Due to the complex methods used to avoid detection and the diversity of languages, it is very difficult to detect and recognize a Webshell. Some typical Webshell samples in the real environment are as shown in Fig. 1.



```
example1:
<%@ Page Language = Jscript %>
<%var/*-/*-*/P/*-/*-*/=/*-/*-*/"e"+"v"+/*-/*-*/
"a"+"l"+"("+"R"+"e"+/*-/*-*/"q"+"u"+"e"/*-/*-*/+"s"+"t"+
"[/*-/*-*/0/*-/*-*/-/*-/*-*/2/*-/*-*/-/*-/*-*/5/*-/*-*/]"+
","+"\""+"u"+"n"+"s"/*-/*-*/+"a"+"f"+"e"+"\""+")";eval
(/*-/*-*/P/*-/*-*/,/*-/*-*/"u"+"n"+"s"/*-/*-*/+"a"+"f"+"e"/*-/*-*/);%>

example2:
< % Runtime.getruntime().exec(request.getParameter("cmd")) % >
```

FIGURE 1. Some typical Webshell samples in the real environment.

Traditional Webshell detection methods are usually based on the feature functions of Webshells. Reference [3] proposed an ASP Webshell search software based on a function library directory. Reference [4] identified a Webshell that contains malicious codes by the optimal threshold values. By judging whether the document uses feature words or not, [5] proposed a Webshell detection method based on the lexical level of the scripting language. However, it was difficult to detect Webshells of confusing encryption type.

With the wide application of machine learning in various fields, some researchers began to apply machine learning to Webshell detection by utilizing the text features of opcode sequences and the common statistical features of PHP (Hypertext Preprocessor) files. Reference [6] constructed four different similarity matrices to detect PHP-type malicious Webshells. Reference [7] combined with FastText and the Random Forest [8], [9] algorithm, and proposed an FRF-WD (Random Forest with FastText) model for PHP Webshell detection. Reference [10] used text features of opcode sequences and common statistical features of PHP files to propose a PHP Webshell detection method based on the RF-GBDT (Random Forest Gradient Boosting Decision Tree)

The associate editor coordinating the review of this manuscript and approving it for publication was Nour Moustafa.

model, which combines the Random Forest [11], [12] and Gradient Boosting Decision Tree [13] algorithms. Although these methods have high detection rates, they are all aimed at PHP-type Webshells. Because these methods make use of the opcode of a particular PHP language, it is difficult to transplant them to Webshells of other language types. Combining the statistical features and special function features of Webshells, [14] proposed an XGBoost [15] algorithm based on statistical features and feature functions. Reference [2] proposed Webshell detection by transforming the statistical features and feature functions of Webshells into matrix form. Although these algorithms can detect all types of Webshells, their accuracy was not high.

Due to the limitation of feature selection, traditional detection methods based on feature functions and machine learning have different disadvantages, thus the classification performance is not good enough. In referring to the way humans recognize Webshells, we consider the neural network methods to learn the Webshells and normal web pages, and then classify them. ''Neural network'' is a mathematical or computational model that imitates the structure and function of a biological neural network. At the same time, through training and learning, a neural network can have better memory ability than humans. Recurrent Neural Network (RNN) [16] has good performance in many NLP (natural language processing) tasks; therefore, this paper considers solving the problem of Webshell detection using RNN.

The problem of Webshell detection can be regarded as the classification of Webshells and common web pages, which is similar to text classification and emotional classification. In addition, they all require semantic analysis. Nowadays, there are many mature algorithms for text classification and emotional classification. Reference [7] averaged all the word vectors in a sentence and then connected them to the softmax layer, and proposed an extremely simple model based on FastText. The influence of word order information on the result was not considered in the FastText network at all. Thus, [17] used a convolutional neural network to extract word order information in sentences and then connected the full connection layer to classify the text. In the process of text categorization and due to the fact that some turning points can change the meaning of the text, it was necessary to retain the global sequence information to improve the detection effect. Reference [18] considered introducing LSTM into text categorization and finally connected the results of the last word to the full connection layer for categorization. Reference [19] analyzed the importance of words and sentences in texts at word-level and sentence-level, and classified them with a full connection layer by combining LSTM and Attention.

However, in order to apply text categorization to Webshell detection, it needs to be optimized and improved. Webshell detection is different from text categorization and there may be semantic reversals, such as turning points between sentences, in text categorization. Therefore, in text categorization, inter-sentence connection information is very important, but the meaning of inter-word connection information in Webshell is higher than that of inter-sentence connection information. This paper presents a Webshell detection model based on the attention mechanism [20]. The model takes the list of arrivals of specific word segments as input after reading the file content by line and then uses Word2vec [21], [22] to vectorize the words.

In Section 2, we review some typical approaches to Webshell classification. Section 3 describes the details of the Webshell classification model using the attention mechanism. In Section 4, an experimental evaluation of classification accuracy is carried out for the proposed model.

II. RELATED WORK

For Webshell detection, common research methods are mainly divided into two aspects: PHP-type Webshell detection and all types of Webshell detection. These are based on machine learning algorithms to classify whether they are Webshells or not. A brief description of each of the methods used in the experiment is given as follows.

The FRF-WD [7] method first uses the sequence characteristics of operation codes to average the word representation into text representation. It then feeds the text representation to the linear classifier to train the FastText model. Finally, the Random Forest model is trained to classify the Webshell by using the results of pre-classification and static features (longest string, information entropy, index of coincidence, signature, blacklist keywords) of FastText model according to the sequence characteristics of operation codes. The RF-GBDT [10] method first extracts text features (a 146-dimensional TF-IDF vector and a 200-dimensional hash vector) from the sequence of PHP file opcodes. Then, it trains an RF classifier based on these text features to obtain preliminary prediction results. Finally, it uses text statistical features (information entropy, index of coincidence, length of longest word, data compression ratio, amount of matched signatures) and preliminary prediction results to train a GBDT classifier and obtain the final classification results. The matrix decomposition method [2] extracts text features (number of words, number of different words, maximum length of words, total length of text, number of notes, number of special characters) and other features (character manipulation function call, key function call, encryption and decryption function call, system function call, file invocation, ActiveX control call, database call, number of script), then groups and combines each feature in all features, and finally using the matrix decomposition model to train and predict. The XGBoost method unifies the performance function, file coincidence index, information entropy, longest string length, and compression ratio as the features of Webshell, and trains the model to classify using the XGBoost algorithm.

III. MODEL ARCHITECTURE

The overall architecture of our model is shown in Fig. 2. It includes four parts: text vectorization, a Gated Recursive Unit (GRU) [23] layer, a word attention mechanism, and dense layers. We first need to convert the Webshells into

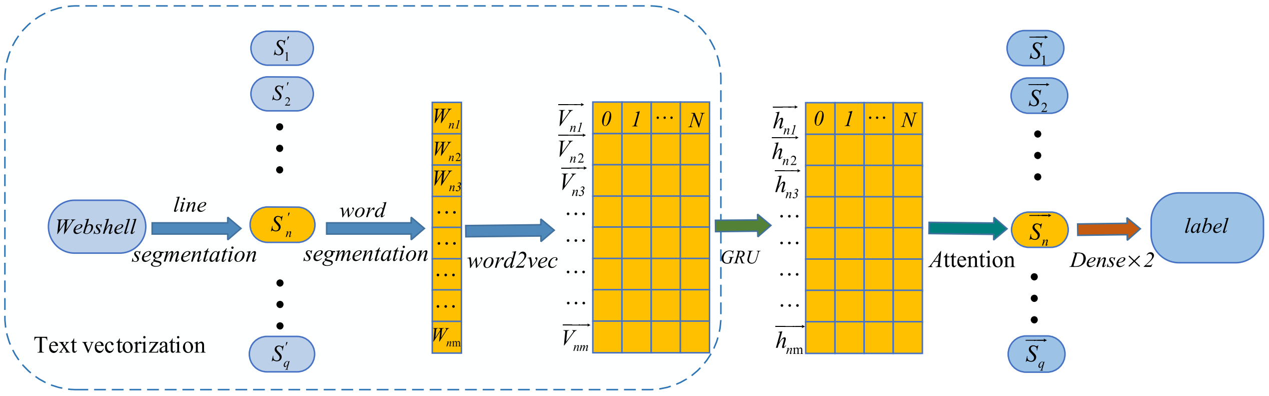
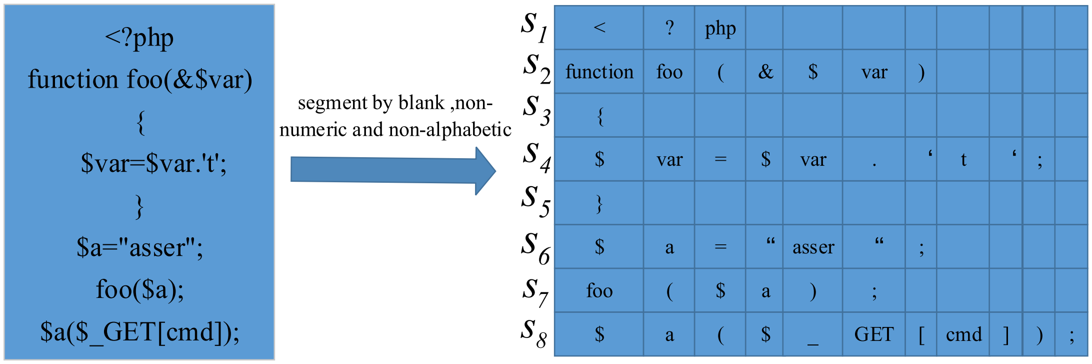


FIGURE 2. Overall architecture of the proposed model.



```
<?php
function foo(&$var)
{
$var=$var.'t';
}
$a="asser";
foo($a);
$a($_GET[cmd]);
```

S_1	<	?	php								
S_2	function	foo	(	&	$	var	)				
S_3	{										
S_4	$	var	=	$	var	.	'	t	'	;	
S_5	}										
S_6	$	a	=	"	asser	"	;				
S_7	foo	(	$	a	)	;					
S_8	$	a	(	$	_	GET	[	cmd	]	)	;

FIGURE 3. Overall architecture of our model.

the vectors commonly used in neural networks through text vectorization. The state of the LSTM at time t is determined by time $t-1$ and the influence of its state before time $t-1$ on the state at time t can only depend on its influence on time $t-1$ [24], [23]. Second, the gated recursive unit (GRU) preserves information in long-term sequences that cannot be erased over time or in relation to predictions. In Webshells, the relationship between words in every line of code is often very important. The semantics of a sentence can be obtained by the information of the relationship between words in a sentence. Third, our model highlights the interaction of words in each line by introducing the word attention mechanism. Finally, we use the dense layers to classify.

A. TEXT VECTORIZATION

According to the language characteristics of the Webshell, the text is segmented as shown in Fig. 3. First, each line of code is treated as a sentence. If words are segmented directly according to the blank space in sentences, then it will lead to more kinds of words, a large vocabulary, and high memory consumption. Moreover, many words with low frequency will be directly filled with vectors when they are vectorized, thus losing their specific meanings and possibly affecting the meaning of sentences. Therefore, in our model, we consider the use of special characters of blank, non-numeric, and non-alphabetic for word segmentation, and the S'_n of the nth sentence is segmented to get list S_n:

$$S_n = [W_{n,1}, W_{n,2}, \cdots, W_{n,m}], \tag{1}$$

where $W_{n,j}$ is the jth word in the nth sentence and m stands for the number of words in a sentence.

After word segmentation, the primary work is to get the vector form of a word. We consider using Word2vec to represent a word as a vector. The vectors obtained by Word2vec vectorization are lower dimension and denser than the one-hot encoder, which uses a vector containing only a 1 and all other 0 to uniquely represent a word. The dimension of vectors obtained by Word2vec is much smaller than the total number of words, but the dimension of vectors obtained by one-hot encoder is generally the same as the number of words. Therefore, Word2vec is essentially a dimension reduction operation. There are many training models for Word2vec and we choose CBOW. CBOW takes the context of vectorized words as input and obtains a word embedding matrix through training. Finally, we use one-hot encoder of words multiplied by word embedding matrix to get word vectors. Therefore, S_n obtained from word segmentation is vectorized by Word2vec to get $[V_{n,1}, V_{n,2}, \cdots, V_{n,m}]$ [21], where $V_{n,j}$ is the vector form of the j_{th} word in the n_{th} sentence.

B. GRU ENCODER LAYER

The Gated Recursive Unit (GRU) proposed by [23] was changed on the basis of long-term and short-term

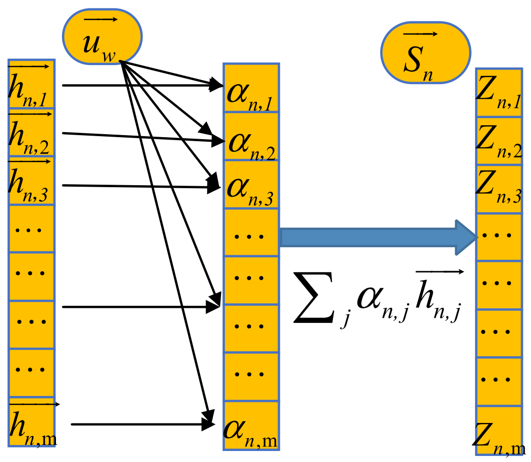

$\overrightarrow{u_w}$
$\overrightarrow{S_n}$
$\overrightarrow{h_{n,1}}$
$\overrightarrow{h_{n,2}}$
$\overrightarrow{h_{n,3}}$
$\overrightarrow{h_{n,m}}$
$\alpha_{n,1}$
$\alpha_{n,2}$
$\alpha_{n,3}$
$\alpha_{n,m}$
$\sum_j \alpha_{n,j} \overrightarrow{h_{n,j}}$
$Z_{n,1}$
$Z_{n,2}$
$Z_{n,3}$
$Z_{n,m}$

FIGURE 4. Architecture of attention layer.

memory (LSTM) as proposed by [25], which includes three gating units. In Gated Recursive Unit, the input gate, and the forgetting gate are merged into an update gate, and the output gate is changed into a reset gate. The overall parameters are reduced. Therefore, the training speed of GRU is slightly faster than LSTM. Update and reset gates in gated recursive units can preserve information in long-term sequences and cannot be cleared due to time lapse or irrelevance to prediction. Therefore, it can effectively avoid the problem of gradient disappearance. The vectors $V_{n,j}$ of the jth word in the nth line are computed through GRU ($j = 1, 2, \cdots, m$) layer to get $h_{n,j}$ of the hidden state, which was computed through $\tilde{h}_{n,j}$ of the candidate activation.

C. WORD ATTENTION MECHANISM

In the process of classification, we need to compress the information of each sentence into a fixed dimension vector. At the same time, not all words in the sentence have the same importance to the meaning of sentence expression, so we add the attention mechanism to the model [20]. The architecture of the attention layer is shown in Fig. 4. At each step of compression, the hidden state is taken as input and the weight representing the importance of the hidden state is calculated at each input position. Then, the vocabulary information in the sentence is added into the sentence vector by the trained weight vector. Therefore, we first get the hidden representation $u_{n,j}$ of the word vector through a single multi-layer neural network and then use the word-level importance vector $u_{n,j}$, which was based on the sentence and u_w, to get the normalized weight $\alpha_{n,j}$ through the softmax function. Finally, the weighted sum vector S_n of the word information is obtained according to the word information $h_{n,j}$ and the normalized weight $\alpha_{n,j}$. The calculation equations are as follows:

$$u_{n,j} = tanh(W_w h_{n,j} + b_w), \tag{2}$$

$$\alpha_{n,j} = \frac{\exp(u_{n,j}^T u_w)}{\sum_j \exp(u_{n,j}^T u_w)}, \tag{3}$$

$$S_n = \sum_j \alpha_{n,j} h_{n,j}, \tag{4}$$

where W_w is a multi-layer neural network parameter, b_w is a bias term, and u_w is a context vector. The hidden state contains the information of the whole target sentence.

D. DENSE

The document matrix composed of the sentence vectors of the file is a high-level representation of the document and can be used as features for document classification. In the model, we use their features to classify whether they are Webshells or not. After obtaining the sentence vectors of the file, a two-dimensional matrix of the file is mapped to a one-dimensional vector through a full connection layer of sigmoid function to reduce the parameters. Finally, we use a sigmoid function full connection layer to classify each file.

IV. EXPERIMENTAL CLASSIFICATION RESULITS ANALYSIS

The code we used to train and evaluate our models is available at https://github.com/leett1/Programe/. The Webshell samples were collected from Github such as https://github.com/tennc/webshell, etc. We also uploaded 500 normal web page samples and 500 Webshell samples for reference.

A. DATASET ENVIRONMENT

We evaluate the effectiveness of our model in classifying Webshells in various language types. PHP, ASP, ASPX, and JSP are used more frequently in Web development; therefore, this paper takes normal web page samples and Webshell samples written in PHP, ASP, ASPX, and JSP. We collect about 7,400 normal web page samples and 4,500 Webshell samples of open source web projects written in PHP, ASP, ASPX, and JSP languages from Github. To obtain different training and testing data, we randomly divide the normal web page samples into 7:3. Then, we use 70% of the samples for training and 30% for testing. Similarly, we randomly divide Webshell samples into 7:3 and use 70% of the samples for training and 30% for testing. Finally, the training samples of common web page samples and Webshell samples are merged into the training set and the testing samples of common web page samples and Webshell samples merged into the testing set. We implement our model in Tensorflow and train it on GTX1060GPU.

B. MODEL TRAINING AND PARAMETER SETTING

We divide each file into sentences by using each line as a unit and then segment the sentences by using the word segmentation method proposed by us to obtain the word sequence of each sentence, thereby encapsulating each file into a matrix and marking each file with a type. When building a vocabulary, we only retain words that appear more than 20 times in the training set. We use the training set to train an unsupervised Word2vec model to get word embedding and then use word embedding to initialize and validate the

TABLE 1. Confusion matrix.

Actual	Predicted	
	Positive	Negative
Positive	TP = 1371	FN=14
Negative	FP=15	TN=2177

training data. The hyper-parameters of the model are optimized on the verification set. In the experiment, we set the embedding dimension to 150, the GRU dimension to 100, and the dimension of the word context vector to 100, which is randomly initialized. In the training process, we set the batch size to 25 and the number of training rounds to 10. At this time, the model training effect and test effect are the best.

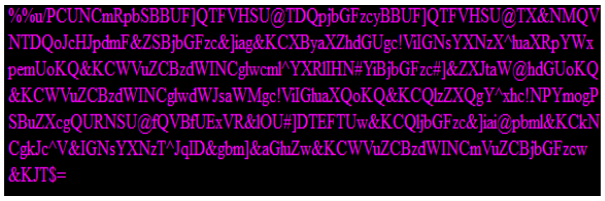
C. EXPERIMENT RESULTS AND ANALYSIS

Webshell samples are labeled as positive and normal web page samples as negative. The confusion matrix presented in Table 1 shows that the TP (True Positive) and TN (True Negative) values are both much higher than for FP and FN. The TP (True Positive) refers to Webshell samples correctly classified by the classifier, the TN (True Negative) refers to normal web page samples correctly classified by the classifier, the FP refers to Webshell samples mistakenly labeled as normal web page samples, and the FN refers to normal web page samples labeled as Webshell samples.

Some typical FP and FN samples are shown in Figs. 5 and 6. On the one hand, it can be seen that there are a few normal web page samples in order to achieve certain functions, so they have been confused with coding. Therefore, our model mistakenly classifies them as Webshell samples. On the other hand, in order to hide Webshell samples, some attackers embed the codes into the image so that when we classify them they will appear as garbled phenomena, resulting in less information. Their codes are embedded at will and may be embedded at the back of a line. Our model is limited by the number of lines and words per line, resulting in no important information. Therefore, we mistakenly classify them as normal web page samples. In the future, we will conduct in-depth research on the problem of error classification mentioned in this paper.

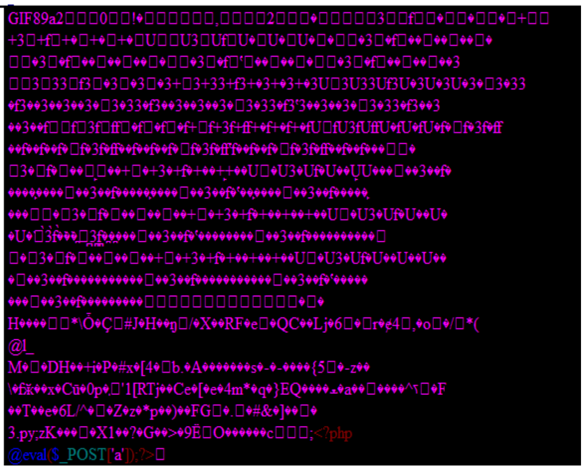
To evaluate the performance of the model accurately, we use the accuracy (ACC), recall rate (Recall), and the F1 score (F1) [26]. Among them, recall rate represents the proportion of positive samples correctly predicted in the test samples, accuracy rate represents the proportion of the whole sample set correctly predicted, and the F1 value combines the results of precision rate and recall rate. The calculation equations are as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = 0.9919, \quad (5)$$



```
%%u/PCUNCmRpbSBBUF]QTFVHSU@TDQpjbGFzcyBBUF]QTFVHSU@TX&NMQV
NTDQoJcHJpdmF&ZSBjbGFzc&]iag&KCXByaXZhdGUgc!ViIGNsYXNzX^luaXRpYWx
pemUoKQ&KCWVuZCBzdWINCglwcml^YXRlIHN#YiBjbGFzc#]&ZXJtaW@hdGUoKQ
&KCWVuZCBzdWINCglwdWJsaWMgc!ViIGluaXQoKQ&KCQlzZXQgY^xhc!NPYmogP
SBuZXcgQURNSU@fQVBfUExVR&lOU#]DTEFTUw&KCQljbGFzc&]iai@pbml&KCkN
CgkJc^V&IGNsYXNzT^JqID&gbm]&aGluZw&KCWVuZCBzdWINCmVuZCBjbGFzcw
&KJT$=
```

FIGURE 5. Typical example of FP.



```
GIF89a2□□□0□□!�□□□□□□□,□□□□2□□□�□□□□□□3□□f□□�□□□�□□□�□+□□
+3□+f□+�□+�□+�□U□□□U3□Uf□U�□U�□U�□�□□�3□�f□��□��□��□�
□□�3□�f□��□��□��□�□□�3□�f□'□��□��□�□□�3□�f□��□��□��3
□□3□33□f3□�3□�3□�3+□3+33+f3+�3+�3+�3U□3U33Uf3U�3U�3U�3�□3�33
�f3��3��3��3�□3�33�f3��3��3��3�□3�33�f3'3��3��3�□3�33�f3��3
��3��f□□f□3f□ff□�f□�f□�f+□f+3f+ff+�f+�f+�fU□fU3fUffU�fU�fU�f�□f�3f�ff
��f��f��f�□f�3f�ff��f��f��f�□f�3f�ff'f��f��f�□f�3f�ff��f��f���□□�
□3�□f�□��□□��+□�+3�+f�+��++��U□�U3�Uf�U��UU���□��3��f�
��������□��3��f���������□��3��f�'������□��3��f�����
���□□�□3�□f�□��□��□��+□�+3�+f�+��+��+��U□�U3�Uf�U��U�
�U�□3f���□3f�����□��3��f�'��������□��3��f�����������□
□�□3�□f�□��□��□��+□�+3�+f�+��+��+��U□�U3�Uf�U��U��U��
�□��3��f�����������□��3��f�����������□��3��f�'�����
���□��3��f���������□□□□□□□□□□□□□□□□�□�
H����□□*\Ō�Ç□#J�H��ŋ□/�X��RF�e□�QC��Lj�6□�□r�ę4□,�o□�/□*(
@l_
M�□�DH��+i�P�#x�[4�□b.�A�������s�-�-����{5□�-z��
\�fx̆��x�Cū�0p�□'1[RTj��Ce�[�e�4m*�q�}EQ����-�a��□����^ٔ□�F
��T��e�6L/^�□�Z�z�*p��)��FG□�.□�#&�]��□�
3.py;zK���□�X1��?�G��>�9Ë□O������c□□□;<?php
@eval($_POST['a']);?>□
```

FIGURE 6. Typical example of FN.

$$Recall = \frac{TP}{TP + FN} = 0.9899, \quad (6)$$

$$Precision = \frac{TP}{TP + FP} = 0.9891, \quad (7)$$

$$F1 = \frac{2PR}{P + R} = 0.9895. \quad (8)$$

Equations (5)–(8) show that the accuracy, recall, precision, and F1 values are about 99%. "Accuracy" can judge the overall accuracy, but it cannot be used as a good indicator to measure the results in the case of unbalanced samples. The recall rate is the ratio of the real positive case to the real positive case in the data, ignoring the influence of erroneous judgement, and is unable to evaluate the model comprehensively. Precision is the ratio of positive predictions in positive samples. F1 is the harmonic average of accuracy and recall. However, by synthesizing the above three parameters, we can see that the detection effect of our training model on each index is very balanced and the effect is very good, and it can distinguish Webshells from ordinary web pages very well.

Figure 7 shows the ROC curve that is a Comprehensive Index of TP and FP. The horizontal axis is the false positive rate (FPR), which is the proportion of common web pages incorrectly classified as Webshell samples and accounts for all common web pages in the test result. The vertical axis is the true positive rate (TPR), which is the proportion of actual Webshells in all Webshell samples. The ideal goal is TPR $= 1$, FPR $= 0$; that is, the closer the area value under the ROC curve is to 1, the higher the accuracy of recognition

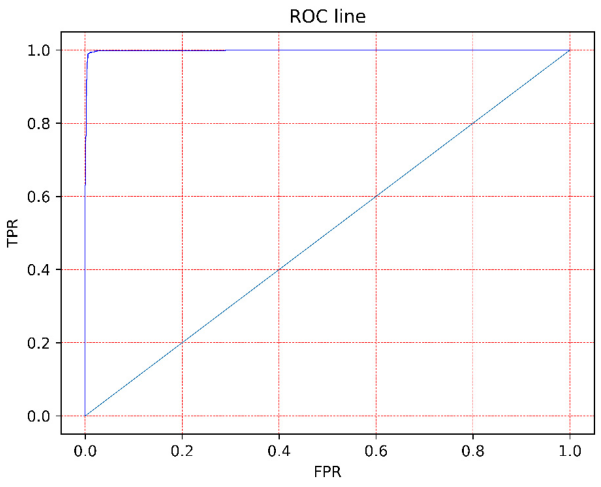


FIGURE 7. ROC curve of the experimental results.

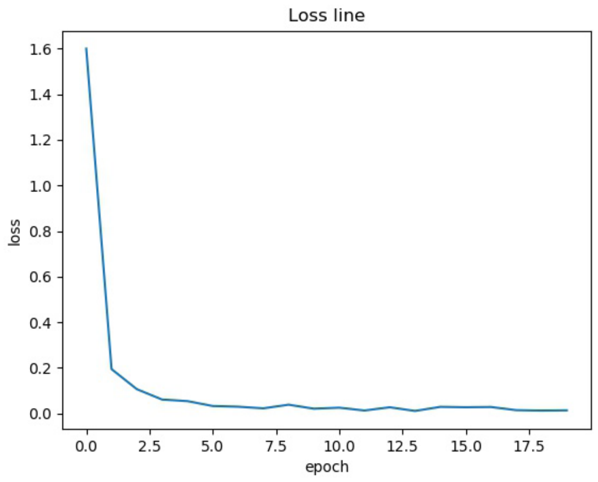


FIGURE 8. Loss value of each epoch.

for Webshells. As can be seen from this, the comprehensive performance of our model is good. Figure 8 shows the loss value of the experimental results of the model. From Fig. 8, we observe that when the epoch value reaches 7, the loss value tends to be stable, about 0.02 in the training process. Hence, one can see that our model has the characteristics of fast convergence.

D. ALGORITHM COMPARISON

We use a confusion matrix as a quantitative evaluation criterion to compare the performance of our model with that of the improved algorithm based on traditional machine learning. The following tables compare machine learning methods for PHP-type Webshell detection and all types of Webshell detection with our model.

From Table 2, we can see that the traditional machine learning methods (RF-GBDT [10], FRF-WD [7]) have better detection rates than our model because extracting some text features from opcode sequences can filter out some noise in the PHP source code, such as some comments. As a result, it is more effective than learning directly from PHP source files. However, although our attention model does not use an opcode, its detection effect is only slightly lower than that of the method of extracting an opcode. However, for all types of Webshell detection, our model detection effect is significantly better than the general machine learning method.

TABLE 2. Comparison of detection effectiveness of PHP type.

Methods	ACC	Recall	F1
RF-GBDT	**0.99169**	0.99093	**0.99093**
FRF-WD	0.9923	0.9765	0.9778
OUR MODEL	0.9894	**0.9967**	0.9847

TABLE 3. Comparison of all types of detection results.

Methods	ACC	Recall	F1
Matrix decomposition	0.982	0.813	0.890
XGBoost	0.8625	08361	0.8619
CNN	0.9729	0.9697	0.9658
TextCNN	0.3931	0.9834	0.5619
LSTM	0.7559	0.5538	0.6373
Our model	**0.9919**	**0.9899**	**0.9895**

For all types of Webshells, we compare existing Webshell detection models based on traditional machine learning (Matrix decomposition, XGBoost) and the classical neural network model. By analyzing and researching the neural network, we put forward several detection schemes based on the neural network. We perform word vectorization on the samples and then use classical neural network models, such as CNN (convolutional neural network) [27], TextCNN [17], and LSTM [25] (long-term and short-term memory network), to train the model. Because we do not extract the unique features of a language during the training and testing processes, our model can be targeted at Webshells of various language types.

Table 3 shows that under the same training and test sets, combined with the accuracy recall and F1 value, the effect of the model based on the GRU attention mechanism is better than that based on other neural networks. Traditional machine learning models only train by extracting some fixed feature functions and statistical features; thus, the insufficient feature library will lead to poor detection results. The model based on CNN can accurately identify local features, but does not consider the location of features, so the effect is not very good. Although the recall rate of the model based on TextCNN is very high, it can only prove that the model can detect the Webshell well and it can also identify many ordinary web pages as Webshells. Therefore, the overall effect is not good.

We use GRU to retain useful information in long sequences and ignore some unimportant information at the same time. Compared with ordinary LSTM, we add the word attention mechanism, which can actively search for the most relevant information at any time while ignoring the irrelevant information; hence, the result of our model is better.

V. CONCLUSION

This paper described the common hidden means and hazards of Webshells and studied traditional machine learning detection methods. Traditional machine learning methods mainly use feature words and statistical features to classify and detect. However, these methods have some shortcomings, such as insufficient feature lexicon and avoiding feature lexicon, and the detection effect depending entirely on the extraction of feature words. Therefore, this paper proposed learning from normal samples and Webshell samples, and then detected Webshell samples according to the learning model. This paper compared the improved algorithm based on traditional machine learning from three aspects: accuracy, recall, and precision. The experimental results show that our model can detect Webshell samples accurately. The detection effect of the model is verified from the aspects of accuracy, recall, precision, and F1 value, and the model effect is evaluated comprehensively by the ROC curve.

REFERENCES

[1] Y. Tian, J. Wang, Z. Zhou, and S. Zhou, "CNN-Webshell: Malicious Web shell detection with convolutional neural network," in *Proc. 6th Int. Conf. Netw., Commun. Comput. (ICNCC)*, 2017, pp. 75–79.

[2] X. Sun, X. Lu, and H. Dai, "A matrix decomposition based webshell detection method," in *Proc. Int. Conf. Cryptogr., Secur. Privacy (ICCSP)*, 2017, pp. 66–70.

[3] X. Mingkun, C. Xi, and H. Yan, "Design of software to search ASP Web shell," *Procedia Eng.*, vol. 29, pp. 123–127, Jan. 2012.

[4] T. Dinh Tu, C. Guang, G. Xiaojun, and P. Wubin, "Webshell detection techniques in Web applications," in *Proc. 5th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, 2014, pp. 1–7.

[5] L. Y. Deng, D. L. Lee, Y.-H. Chen, and L. X. Yann, "Lexical analysis for the webshell attacks," in *Proc. Int. Symp. Comput., Consum. Control (IS3C)*, 2016, pp. 579–582.

[6] P. M. Wrench and B. V. W. Irwin, "Towards a PHP Webshell taxonomy using deobfuscation-assisted similarity analysis," in *Proc. Inf. Secur. South Africa (ISSA)*, 2015, pp. 1–8.

[7] Y. Fang, Y. Qiu, L. Liu, and C. Huang, "Detecting webshell based on random forest with FastText," in *Proc. Int. Conf. Comput. Artif. Intell. (ICCAI)*, 2018, pp. 52–56.

[8] J. Armand, "Bag of tricks for efficient text classification," in *Proc. EACL*, 2017.

[9] A. Liaw and M. Wiener, "Classification and regression by random forest," *R News*, vols. 2–3, pp. 18–22, Dec. 2002.

[10] H. Cui, D. Huang, Y. Fang, L. Liu, and C. Huang, "Webshell detection based on random Forest–Gradient boosting decision tree algorithm," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2018, pp. 153–160.

[11] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, Montreal, QC, Canada, vol. 1, 1995, pp. 278–282.

[12] L. Breiman, "Arcing the edge," Dept. Statist., Univ. California at Berkeley, Berkeley, CA, USA, Tech. Rep. 486, 1997.

[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, pp. 1189–1232, Oct. 2001.

[14] C. Yanpeng, S. Kexing, and H. Jianwei, "Research of Web shell detection method based on XGBoost," *Comput. Sci.*, vol. 6A, pp. 375–379, Jun. 2018.

[15] *A Gentle Introduction to XGBoost for Applied MachineLearning [EB/OL]*. Accessed: Aug. 2016. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[16] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016, *arXiv:1605.05101*. [Online]. Available: https://arxiv.org/abs/1605.05101

[17] K. Yoon, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, 2014.

[18] L. Pengfei, "Recurrent neural network for text classification with multi-task learning," *IJCAI*, 2016.

[19] Y. Zichao, "Hierarchical attention networks for document classification," in *Proc. HLT-NAACL*, 2016.

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2015, *arXiv:1409.0473*. [Online]. Available: https://arxiv.org/abs/1409.0473

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781

[22] T. Chen, Q. Mao, M. Lv, H. Cheng, and Y. Li, "DroidVecDeep: Android malware detection based on Word2Vec and deep belief network," *TIIS*, vol. 13, no. 4, pp. 2180–2197, 2019.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," Dec. 2014, *arXiv:1412.3555*. [Online]. Available: https://arxiv.org/abs/1412.3555

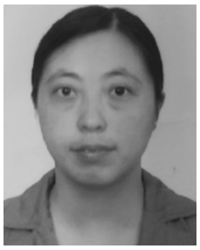
[24] A. Graves, "Supervised sequence labelling with recu works," in *Studies in Computational Intelligence*. Berlin, Germany: Springer, 2008, doi: 10.1007/978-3-642-24797-2.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[26] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *Markedness Correlation*, vol. 2, no. 1, pp. 37–63, 2011.

[27] K. Simonyan and A. Zissersman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

TINGTING LI received the bachelor's degree in communication engineering from Sichuan Normal University, Chengdu, China, in 2017. She is currently pursuing the degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China. Her current research interests include natural language processing and the Internet security.



CHUNHUI REN received the bachelor's, master's, and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, in 1992, 1998, and 2006, respectively. She is currently an Associate Professor of communication and information engineering, University of Electronic Science and Technology of China. In recent years, her research has mainly focused on electronic countermeasures, statistical signal processing, non-cooperative signal processing, and other fields. She has published more than ten academic articles, including several articles in SCI (including SCIE) source journals and several EI-retrieved journals and conference papers.

YUSHENG FU received the bachelor's degree in avionics engineering from Air Force Engineering University, Xi'an, China, in 1995, and the master's and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, in 2000 and 2004, respectively. He is currently an Associate Professor of communication and information engineering, University of Electronic Science and Technology of China. His research in the past five years has mainly focused on signal processing, aero-electronics, and biomedical electronics engineering, and he has conducted recent research on network science and technology. Over the past five years, a total of ten million yuan has been spent on scientific research, with an average annual expenditure of more than one million yuan. He has published more than ten academic articles, including three SCI articles and more than ten EI articles. He is a Reviewer of Circuits, Systems, and Signal Processing and other academic journals.

JIE XU received the bachelor's degree from Chongqing University, in 2003, the master's degree from the Université de Franche-Comté, in 2004, and the Ph.D. degree from the Institut National des Sciences Appliquées (INSA - Toulouse), in 2008. He is currently a Lecturer and an Associate Professor with the University of Electronic Science and Technology of China. In recent years, his research has mainly focused on the fields of network and information security, video recognition and security, data mining and analysis, and chaotic dynamics and applications. His research results have been published in more than 30 articles in domestic and foreign academic journals and international academic conferences, and he has applied for more than 20 national patents for technological inventions. He is a member of the Procedure Committee (TPC) of academic conferences. He serves as a Reviewer of journals in China and abroad.

JINHONG GUO received the bachelor's degree in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, in 2010, and the Ph.D. degree in biomedical engineering from Nanyang Technological University, in 2014. He is currently a Full Professor with the School of Communication and Information Engineering, University of Electronic Science and Technology of China, and the Chengdu University of Traditional Medicine. After his doctoral studies, he was a Postdoctoral Fellow with the Pillar of Engineering Design, MIT-SUTD, Singapore, from 2014 to 2015. He then worked as a Visiting Professor with the School of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA, from January 2016 to July 2016. His current research focuses on electrochemical sensors and lab-on-a-chip devices for point-of-care clinical testing. He was a recipient of the China Sichuan Thousand Talents Plan for Scholars Award, in 2015, and the Chengdu Expert in Science and Technology Award, in 2015. He is also appointed as the Chief Scientist at Longmaster Information Company, Ltd., (one listed corporation in China, stock ID: 300288) and is in charge of the research and development center for POCT. He has published over 70 articles in high-impact journals, such as the IEEE TII, TBME, TBioCAS, Analytical Chemistry, Biosensor, and Bioelectronics.

XINYU CHEN is currently a Lecturer of communication and information engineering, University of Electronic Science and Technology of China. He has been involved in the research of digital audio–visual signal processing and broadcasting engineering technology for a long time. Products such as magneto-optical disc recorder and Cobranet audio router, in which his research group participated in, have been awarded at the provincial and ministerial levels. They have been widely applied in the broadcasting industry and have obtained good evaluation. He has published many articles in core professional journals in China.